AFOSR-TR-95

REPORT DOCUMENTATION PAGE

0660

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		FINAL/15 JUL 91 TO 14 FEB 95

4. TITLE AND SUBTITLE
FAST MULTIPOLE METHODS FOR SCATTERING COMPUTATION

5. FUNDING NUMBERS
7894/08
F49620-91-C-0064

6. AUTHOR(S)
L. R. HAMILTON, J. J. OTTUSCH, R. S. ROSS, M. A. STALZER, R. S. TURLEY, J. L. VISHER, S. M. WANDURA

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
HUGHES RESEARCH LABORATORIES
3011 MALIBU CANYON ROAD
MALIBU, CALIFORNIA 90265

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
110 DUNCAN AVE, SUTE B115
BOLLING AFB DC 20332-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
F49620-91-C-0064

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

At the outset of this contract, the focus of attention was the fast multipole method (FMM) for the Maxwell equations. The FMM essentially provides for a fast and accurate computation of "non-near" (in a sense made precise in [CRW93a]) fields of specified currents. The techniques [Rok90, Rok93, CRW93a] of FMM's were pioneered at Yale University by Professor Vladimir Rokhlin. During our contract work, it became clear that the problem of efficient and accurate computation of near field effects was as challenging and needy of study as the FMM. By the time the contract was completed, we had made significant and fundamental progress on both fronts.

19951018 013

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR(SAME AS REPORT)

HAC REF J3480

FAST MULTIPOLE METHODS FOR SCATTERING COMPUTATION

Hughes Research Laboratories

3011 Malibu Canyon Road

Malibu, California 90265

L.R. Hamilton
J.J. Ottusch
R.S. Ross
M.A. Stalzer
R.S. Turley
J.L. Visher
S.M. Wandzura

September 1995

Final Report

Contract No. F49620-91-C-0064

July 15, 1991 Through February 14, 1995

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
110 Duncan Avenue, Suite B115
Bolling Air Force Base, DC 20332-0001

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 7894
Monitored by AFOSR Under Contract No. F49620-91-C-0064

Contents

Chapter 1

Summary

1.1 Introduction

Conventional methods for computing radar cross section (RCS) are limited by computer processing time and memory requirements. These constraints limit the practical scale size for accurate predictive codes to about 10 wavelengths, far below the actual size of interest for most military platforms. We have demonstrated fast, high-order computational techniques that dramatically reduce the computation time and memory for large scale RCS calculations. When implemented on large computers, this will enable accurate RCS prediction for problems that are currently intractable. This development will clearly be of tremendous advantage for design of future low observable platforms, where previously one could only rely upon expensive experimental measurements.

At the outset of this contract, the focus of attention was the fast multipole method (FMM) for the Maxwell equations. The FMM essentially provides for a fast and accurate computation of "non-near" (in a sense made precise in [CRW93a]) fields of specified currents. The techniques[Rok90, Rok93, CRW93a] of FMM's were pioneered at Yale University by Professor Vladimir Rokhlin. During our contract work, it became clear that the problem of efficient and accurate computation of *near* field effects was as challenging and needy of study as the FMM. By the time the contract was completed, we had made significant and fundamental progress on both fronts.

The work carried out under this contract has been done in concert with work done under a parallel contract (F49620-91-C-0084) to the Fast Mathematical Algorithms and Hardware Corporation (FMAHC), which supports mathematical research by Professors Rokhlin and Ronald Coifman on FMM and wavelet techniques for scattering calculations. Hughes has an ongoing IR&D program in areas which are closely related to the work we are pursuing under this contract. The IR&D results significantly enhance the value of the items delivered under

this contract and are made available to the Government under limited rights.

Having worked out (under the guidance of FMAHC) the theoretical details of the $3d$ FMM in the first year, we went on to implement the FMM for $2d$ and $3d$ scalar and electromagnetic scattering. Efficient implementation for the $3d$ Neumann and electromagnetic cases required some additional technical details. We made valuable progress on the theory of numerical integration necessary for high-order discretization of boundary integral equations and for the $3d$ FMM. We found high order Gaussian-type quadrature rules for integrals over triangles and spheres. We also found Gaussian quadrature formulas for integration over an interval of linear combinations of regular functions and functions with logarithmic singularities at one or both endpoints. Such rules are useful for efficient, accurate numerical solution of boundary integral equations that have singular kernels, such as that for the wave equation. Near the end of the contract, we devised and implemented an entirely new approach to high-order discretization of field (both differential and integral) equations: the high-order regulation of singular kernels. This technique can be employed with either Galerkin or Nyström discretizations. Under our IR&D project, the design and implementation of our object-oriented scattering code, FastScat* has been a success. Although it took six months of coding to compute scattering from the simplest surface (polygons in $2d$ with pulse basis functions), the speed with which we have been able to implement curvilinear surface modeling, higher-order basis functions, new iterative solvers, the FMM, and the extension to $3d$ electromagnetic scattering validates our implementation approach. In the following, we discuss both our internal and Government sponsored research.

1.2 Results

We report our progress in four areas:

- High-order techniques for method of moments (MoM) discretization of integral equations;
- Various techniques for FMM implementation;
- The design and implementation of FastScat, Hughes' object-oriented scattering code;
- FastScat testing and validation.

The first two were supported by this contract. Most of the FastScat development was supported by Hughes IR&D, with the exception of the $2d$ curvilinear patch work and $3d$ FMM implementation, which was funded under the contract.

*FastScat is a trademark of Hughes Aircraft Company.

1.2.1 Discretization

A principal and solitary achievement during this contract was the design and construction of a high-order discretization for three dimensional surface scattering problems. The challenge of this was grossly underestimated at the start of the contract. We tried (and abandoned) several approaches as documented in our annual reports. Even the discarded techniques had some valuable byproducts, most notably the generalization of Gaussian quadratures to combinations of singular and nonsingular functions. The approach finally adopted seems clearly the simplest and most efficient. Our discretization approach is presented in terms of the Galerkin technique; however, most of the issues that must be addressed and their appropriate resolution apply to Nyström discretization schemes as well.

Basis Functions

The integral equations pertaining to interesting scattering problems typically involve differential as well as integral operators. Because of this, conventional Galerkin discretizations often employ basis functions that have some automatically enforced degree of continuity. Construction of appropriately continuous basis sets is in fact a major component of the finite element enterprise. We expended considerable effort constructing such bases of arbitrary order. In the end, we changed to a technique that obviates the whole exercise and lets one construct well conditioned bases very easily.

Quadratures

We made fundamental progress in numerical integration, developing new quadrature rules for singular functions on intervals and nonsingular functions on triangles and spheres.

Kernel Regulation

Near the end of the contract, we constructed a radical approach to high-order discretization of integral equation: the substitution of a carefully chosen regular kernel for normal singular kernel. The key features of the modified kernel are that it differs from the true kernel only locally and that it gives results identical to the true kernel for convolutions with a suitable set of functions. The effort was motivated in part as a way to incorporate the local corrections needed for a high- order Nyström method directly into the kernel. It turns out that this is not possible for general geometries without excessive sampling, but the new method can be used to subsume the geometrical information in the precomputation stage.

1.2.2 Fast Multipole Theory

Understanding of the FMM for the Helmholtz equation has grown rapidly during the course of the contract, first from the FMAHC-Hughes collaboration, followed by significant work at Boeing and the University of Illinois. Besides the generalization of Rokhlin's original $2d$ method to higher dimensions, the most important developments have been the further compression of the translation operator and detailed understanding of the higher dimensional versions of the multilevel FMM.

1.2.3 FastScat Code Development

We developed the FastScat program for accurately computing scattering and radiation from $2d$ or $3d$ surfaces of arbitrary shape and user-specified boundary conditions. The program allows the accurate specification of the surface model through the use of a variety of flat and curved patches, and supports the use of higher order basis functions. The linear operator of the integral equation can be represented in various ways, such as conventional dense of FMM. (In the future special representations for symmetrical bodies will be accommodated.) The method of solution can be specified as direct, or one of a number of iterative techniques, for example the biconjugate gradient method.

FastScat was developed using an object-oriented methodology, and was written primarily in C++. This methodology makes implementation modifications involving data representation much less costly than is the case with a conventional procedural approach. FastScat is the only program of its kind — combining high order methods, curved surface models, and the FMM. It was one of the first large numerics programs written in C++, and remains a pioneer in this area.

1.2.4 FastScat Results

We verified that FastScat computes accurate answers by comparing its results to series solutions, measured data, and other programs. We also verified that the program exhibits the high-order convergence expected of the algorithms used. As a result of this high-order convergence, we are able to get accurate estimates of our computational accuracy, without expending excessive resources.

We have compared the solution times and memory requirements of FastScat with the EMCC benchmark programs RAM2D and CARLOS-3D. For $2d$ problems larger than a wavelength, and requiring at least one significant digit of accuracy, FastScat is faster than RAM2D and uses less memory. For $3d$ problems, FastScat has faster solution times and uses less memory than CARLOS-3D. For most problems, when demanding only a few digits of precision, CARLOS-3D is still faster at computing matrix elements since this portion of the FastScat code has not yet been optimized.

Chapter 2

Discretization

2.1 Introduction

Discretization refers to the process of approximating a problem concerning functions (physically, fields or sources) having an infinite number of unknowns by a problem with a finite number of unknowns. This process is obviously essential to the numerical solution of field problems. It is a fundamental tenet of Professor Rokhlin's approach to computational physics that discretizations must be high order. High order means that the error in a computation decreases rapidly (like a high power or, better yet, exponentially) as the number of parameters used to describe a function increases. The value of high-order discretizations is that they allow economical error control and estimation[Ott94, HOS+95a, HOS+95b].

When this contract began, there were (to our knowledge), *no* high-order codes for solution of $3d$ electromagnetic integral equations. Codes other than FastScat (for example, Patch[JWS88], CARLOS[PMMG92], and FERM[LSL87]) compute convolutions with the singular kernel (Green function) in an inherently low-order fashion[Wan95b, Wan95a]. Essentially, they subtract from the kernel a piece that renders the remainder finite but still singular. The subtracted part is integrated analytically; the problem is that Gaussian quadrature is used to approximate the still singular remainder. These codes usually have other features that restrict them to low-order computation, such as crude surface models and low-order basis functions.

In the course of this contract work, we developed and implemented two approaches to high-order discretization of integral equations with singular kernels. Both approaches were implemented and are discussed here in the context of a Galerkin (method of moments) approach to discretization; the elements of each also apply to a Nyström method. The first approach uses quadratures that are specially constructed to give high-order results for the particular singularity involved. The second substitutes a nonsingular (*regulated*) kernel for the singular

kernel. The regulation procedure is dictated by the requirements of locality of correction and preservation of convolutions with a suitable class of smooth functions.

2.2 Basis Functions

Constructing a high order discretization of a surface scattering problem in three dimensions is fundamentally more difficult than in two. In two dimensions (for simply connected surfaces), one may adopt a single regular parameterization. A common choice which works well is to parameterize by arc length along the perimeter. In three dimensions, a single regular parameterization is not generally available (consider maps of the earth's surface). One must therefore introduce a set of parameterizations that cover the surface. We will refer to an area sharing a common parameterization as a "patch". Then, when one constructs a discretization, either Galerkin or Nyström, some basis functions must be used on each patch. This is manifest in the case of the Galerkin method, in the case of Nyström, the basis functions are needed to do the precomputation of "local corrections" to the quadrature rules[Str94]. [The local corrections are necessitated by the short distance singularity of the integral equation kernel.]

In our work, we have used simple basis functions consistent with high-order solution convergence. These are polynomials in the patch parameters combined with suitable functions of derivatives of the patch map $\mathbf{x}(u)$. The derivatives are used to achieve good conditioning, as well as provide tangential basis functions for vector problems.

2.2.1 Basis Functions with Enforced Continuity

During most of the contract, we used a formulation where tangential surface derivatives acted on the basis functions. (The alternative is for the derivatives to act on the kernel, which exacerbates the singularity, making high-order integration more difficult.) In order to get finite matrix elements, this requires some degree of continuity to be enforced. In the case of scalar scattering with Neumann boundary conditions (BCs), the functions must be continuous. In two dimensions, for example, this is accomplished by using "rooftop" basis functions supported on adjacent patches in combination with quadratic and higher-order polynomials that vanish at the endpoints of single patches. In the case of electromagnetic scattering it means functions with finite divergence. A method to achieve finite divergence on curvilinear patches was described in [Wan92]. A systematic generalization of this method to arbitrary order is described in [HMS+94a]. Details of the implementation for the Neumann and vector problems are given in Appendix C.

2.2.2 Basis Functions without Enforced Continuity

Near the completion of the contract, we adopted a radically different approach to high-order discretization: the method of high-order regulated kernels (see Section 2.4 following). As this method replaces the singular kernel with a regular one, it is then possible to use a formulation where the tangential surface derivatives operate on the kernel instead of the basis functions. This greatly simplifies the choice of basis functions and the implementation of the matrix fill. For example, it is then easy to construct orthonormal basis functions. Details of the functions we use in FastScat are given in Appendix C; in this section we simply exhibit the basis functions and the recursion relations used to compute them. (Some implementation efficiency is gained by using the same quadrature rule for all matrix elements between a given patch pair; the values for all basis functions at each abscissa can be generated by the recursion relation.)

2d

Each patch is parameterized by a function $\mathbf{x}(u)$, where $0 \leq u \leq 1$. The Jacobian of the map is

$$\sqrt{g(u)} = \left| \frac{d\mathbf{x}}{du} \right| . \tag{2.1}$$

Orthonormal (with respect to dx) basis functions are thus given by

$$f_m(u) \equiv \frac{\sqrt{2m+1}}{g^{1/4}(u)} P_m(2u-1) , \tag{2.2}$$

where P_m is a Legendre polynomial and the patch index is suppressed. The set of Legendre polynomials is computed for each argument by using the upward recursion

$$P_{n+1}(z) = \frac{1}{n+1} \left[(2n+1) z P_n(z) - n P_{n-1}(z) \right] , \tag{2.3}$$

with

$$P_0(z) = 1 \tag{2.4}$$

$$P_1(z) = z . \tag{2.5}$$

3d

Scalar Case FastScat uses curvilinear triangular patches, parameterized by u_1, u_2, where $0 \leq u_{1,2}$ and $u_1 + u_2 \leq 1$. Orthogonal polynomials over triangles are more easily expressed in terms of the variables

$$\xi \equiv \frac{1}{2} (2 - 3u_1 - 3u_2) \tag{2.6}$$

$$\eta \equiv \frac{\sqrt{3}}{2} (u_1 - u_2) . \tag{2.7}$$

[These are called x and y in Appendix C; the Greek symbols are used here to avoid confusion with the components of the map $\mathbf{x}(u_1, u_2)$.] The orthonormal basis functions are

$$f_{nm}(\xi,\eta) \equiv \frac{(2/3)^m\sqrt{(n+1)(2m+1)}}{g^{1/4}} R_{nm}\left(\frac{4\xi-1}{3}\right)(1-\xi)^m P_m\left(\frac{\sqrt{3}\eta}{1-\xi}\right), \tag{2.8}$$

where

$$R_{nm}(\zeta) \equiv P_{n-m}^{2m+1,0}(\zeta), \tag{2.9}$$

and $P_n^{\alpha,\beta}(\zeta)$ is a Jacobi polynomial[AS72]. [Because of the $(1-\xi)^m$ the f_{nm} are indeed polynomials in ξ, η or u_1, u_2 in spite of the argument of the Legendre polynomial.] Presently, FastScat computes the the R polynomials using

$$R_{nn}(\zeta) = 1, \tag{2.10}$$

and the recursion

$$R_{nm}(\zeta) = \frac{(n+m+2)(2n+1)(1-\zeta)^2 R_{n,m+1}(\zeta) - 8(m+1)(n+m+1)R_{n-1,m}(\zeta)}{2(n-m)\left[(2n+1)\zeta - (2n+4m+5)\right]}. \tag{2.11}$$

This is marginally stable, but adequate for moderate ($n \leq \sim 12$) order. For higher order, a more stable computation could be achieved by using the three term recursion

$$R_{n,0}(\zeta) = \frac{\left[1+(2n-1)(2n+1)\zeta\right]R_{n-1,0}(\zeta) - (n-1)(2n+1)R_{n-2,0}(\zeta)}{(n+1)(2n-1)}, \tag{2.12}$$

with $R_{1,0}(\zeta) = (1+3\zeta)/2$ to compute the the $R_{n,0}(\zeta)$, and a three term recursion for $R_{n,m-1}, R_{nm}, R_{n,m+1}$ to generate a tridiagonal system for all R_{nm} for each n. The other BC for this system is supplied by Eq. (2.10).

Vector Case The current FastScat implementation simply uses the the scalar basis functions of the previous section multiplied by the unit tangent vectors

$$\hat{t}_1 \equiv \frac{d\mathbf{x}/du_1}{|d\mathbf{x}/du_1|} \tag{2.13}$$

$$\hat{t}_2 \equiv \hat{n} \times \hat{t}_1, \tag{2.14}$$

where $\hat{n}$ is the unit surface normal. These obviously form an orthonormal basis.

2.3 Quadratures

A high-order discretization of an integral equation is fundamentally based on numerical quadrature. A large fraction of the contract effort was expended

in search of efficient and accurate quadrature techniques applicable to the $3d$ scattering problem. As was the case with respect to the basis functions (see Section 2.2 preceding) the change to the high-order regulated kernel obviated much of the effort spent earlier. In particular, much effort was spent on finding special quadrature rules for integration on real intervals of functions with logarithmic singularities at one or both endpoints, and on variable transformations that allowed these rules to the computation of $3d$ matrix elements. The latter exercise seems to be of ephemeral interest and was documented in [HST+93c].

All quadrature rules found under this contract are tabulated in Appendix D, along with *Mathematica* code that demonstrates their use. This voluminous appendix will not be furnished with all copies of this report, copies (printed or electronic) are available on request.

2.3.1 Tools for Derivation of Quadrature Rules

A quadrature rule is a prescription for approximation to an integral by a weighted sum of function values:

$$I[f] \equiv \int dx\, f(x) \approx \sum_{n=1}^{N} w_n f(x_n)\,. \tag{2.15}$$

The weights w_n and abscissae x_n obey the condition that a set of M functions $f_m(x)$ are integrated exactly:

$$I_m \equiv I[f_m] = \sum_{n=1}^{N} w_n f_m(x_n) \quad ; \quad m = 1, \ldots, M\,. \tag{2.16}$$

These functions are chosen to so that the quality of an approximation to the desired integrand by a linear combination of them increases rapidly with M. The (nonlinear) equations satisfied by the weights and abscissae are very poorly conditioned. (This is the converse of the fact that the computation of the integral by the quadrature rule is very well conditioned.) In this subsection, we exhibit a couple of techniques that have proven useful for the computation of quadrature rules for both singular and nonsingular functions.

Newton-Rokhlin Method

This is an iterative technique to refine a sufficiently good initial guess $(\tilde{w}_n, \tilde{x}_n)$ for a quadrature rule. It results from the straightforward linearization of Eq. (2.16):

$$x_n \approx \tilde{x}_n + \delta x_n \tag{2.17}$$

$$w_n \approx \tilde{w}_n + \delta w_n \tag{2.18}$$

$$I_m - \sum_{n=1}^{N} \tilde{w}_n f_m(\tilde{x}_n) \tag{2.19}$$

$$= \sum_{n=1}^{N} f_m(\tilde{x}_n)\delta w_n + \tilde{w}_n f'_m(\tilde{x}_n)\delta x_n \quad ; \quad m = 1, \ldots, M\,. \tag{2.20}$$

For multidimensional quadrature, there can sometimes be fewer equations than unknowns. In such a case, one can either use a pseudoinverse or enforce a subsidiary condition, such as minimization of $\sum_n w_n^2$. A more complicated exposition of this iterative method (which has the advantage of providing for various interesting proofs) can be found in [MRW93]. Because of the poor condition of these equations, it is necessary to use extended precision arithmetic for their solution in order to obtain ordinary precision in the results.

Simulated Annealing

In order to use the preceding iterative technique, one must start from a sufficiently close initial guess. For fairly high-order rules, the domain of attraction of the iteration is usually uncomfortably small. To help find an initial guess that is in this domain, we have employed the technique of simulated annealing[KGV83, Kir84] on the objective function

$$J = \sum_m \omega_m \left[I_m - \sum_n w_n f_m(x_n) \right]^2 , \tag{2.21}$$

where the ω_m are positive weights. The choice of these weights is an art which can strongly influence the ease of finding rules.

2.3.2 Singular Functions on Intervals

For high-order computation of near* matrix elements of the Helmholtz kernel on a curvilinear surface, it is convenient to be able to compute integrals of the form

$$\int_0^1 dx\, [\psi(x) \ln x + \phi(x)] \tag{2.22}$$

(where ψ and ϕ are regular) without having to separate the two terms in the integrand. In the course of this contract we found that quadrature rules that are exact for polynomial ψ and ϕ (of the same degree) with good behavior (positive weights and abscissae in the integration interval) are available. An analysis of the generalization of these to a wide class of singularities is given in [MRW93]. Rules for integrands of the above form and also with logarithmic singularities at both endpoints are tabulated in Appendix D.

*Those which probe the singularity of kernel at vanishing separation

2.3.3 Regular Functions on Multidimensional Domains

By using elementary group representation theory, one may reduce the number of equations to be solved for the weights and abscissae for integration over symmetric regions. This was detailed in the annual report[HST+93c]. We used these methods to develop rules up to order 29 for triangles and up to order 48 for spheres. The triangle rules use the dihedral group D_3 and the sphere rules use the octahedral group O_h. The triangle rules are used in FastScat because we use triangular patches, the sphere rules are fundamental to the FMM. These rules are also tabulated in Appendix D.

2.4 High-Order Regulation of Singular Kernels

2.4.1 Local Correction of Quadratures by Regulation of Kernels

An alternative to using special quadratures to compute convolutions with singular kernels is the modification of the kernel itself to truly remove the singularity. "Truly" means here to render the kernel *and all its derivatives* finite. Short-distance regulation of kernels is a time-honored technique in physics; it was the key to solving the problem of extracting predictions from higher-order computations in quantum electrodynamics[PV49, Fey49]. (The regulation methods cited were, however, in present terminology, "low-order" — the resultant kernels, although finite for vanishing separations, were still singular.)

The question of how to preserve accuracy control while regulating the kernel is illuminated by the observation that achieving high-order convergence by use of special quadratures, constructed to converge rapidly when integrating something with the singularity structure of the kernel, still requires that the functions that are convolved with the kernel be regular. (If one were to introduce singular basis functions, for example, to model sources near geometric singularities of the scatterer, this approach would need to be modified to take into account the singular behavior of the basis functions.) Thus the key is to require that the regulated kernel not only converge to the singular kernel as the regulation is removed, but that it gives *identical* results to the singular kernel when convolved with a suitable class of smooth functions.

2.4.2 Regulated Kernels for Laplace Equation

2d

We will show how this can be done to the kernel

$$G(r) = \ln r\,, \tag{2.23}$$

for the Laplace Equation in two dimensions for convolution with functions on smooth curves (smooth surfaces of $2d$ objects).

The first step is to Fourier transform to momentum space, giving $(\ln q + \gamma - 1)/q^2$, where γ is the Euler constant. We then regulate the short-distance (large q) with a factor $\exp -q^2/(4\alpha)$. Fourier transforming back to real space gives the regulation prescription

$$\ln r \to \ln r + \frac{1}{2} E_1(\alpha r^2)\,, \tag{2.24}$$

where E_1 is the exponential integral[AS72]. This regulated kernel obviously is nonsingular (because $E_1(x) + \ln x$ is regular), approaches the unregulated kernel (for fixed r) as $\alpha \to \infty$, and has only local corrections. The last property means that the difference between the kernels vanishes as fast as a Gaussian for $r \gg a = 1/\sqrt{\alpha}$, Because of this, we need not concern ourselves with kernel modification for sufficiently far interactions, allowing compatibility with the FMM.

The regulation so far is low order, in the sense that results computed with the regulated kernel will have regulation errors that are proportional to a low power of a. What we want to do, then, is to add a smooth local function that renders the regulation high order. A fairly obvious choice is thus

$$\bar{G}_B(r) = \ln r + \frac{1}{2} E_1(\alpha r^2) - e^{-\alpha r^2} F_B(\alpha r^2) \tag{2.25}$$

$$\bar{G}_B(0) = -\frac{1}{2}(\gamma + \ln \alpha) - F_B(0)\,, \tag{2.26}$$

and F_B is a Bth order polynomial

$$F_B(x) = \sum_{l=0}^{B} f_l^{(B)} x^l\,, \tag{2.27}$$

with coefficients adjusted to enforce

$$\int_0^\infty dr\, r^{2m} \left[\bar{G}_B(r) - G(r)\right] = 0 \quad ; \quad m = 0, \ldots, B \tag{2.28}$$

using

$$\frac{1}{2}\int_0^\infty dr\, r^{2m} E_1(\alpha r^2) = \frac{1}{2\alpha^{m+1/2}} \frac{\Gamma(m+1/2)}{2m+1} \tag{2.29}$$

$$\alpha^l \int_0^\infty dr\, r^{2m+2l} e^{-\alpha r^2} = \frac{1}{2a^{m+1/2}} \Gamma(m+l+\frac{1}{2})\,. \tag{2.30}$$

Thus

$$\int_0^\infty dr r^{2m} \left[\bar{G}(r) - G(r)\right] = \frac{1}{2\alpha^{m+1/2}} \frac{\Gamma(m+1/2)}{2m+1} - \frac{1}{2\alpha^{m+1/2}} \sum_{l=0}^{B} \Gamma(m+l+1/2) f_l^{(B)}\,, \tag{2.31}$$

so that

$$\sum_{l=0}^{B} \Gamma(m+l+1/2) f_l^{(B)} = \frac{\Gamma(m+1/2)}{2m+1} . \tag{2.32}$$

3d

The regulation of the $3d$ Laplace kernel proceeds similarly, the kernel

$$G(r) = \frac{1}{r} \tag{2.33}$$

having the Fourier transform $1/q^2$. Then

$$\bar{G}_B(r) = \frac{\mathrm{erf}\,(\sqrt{\alpha} r)}{r} + \sqrt{\alpha} e^{-\alpha r^2} F_B(\alpha r^2) \tag{2.34}$$

$$\bar{G}_B(0) = \sqrt{\alpha} \left(\frac{2}{\sqrt{\pi}} + F_B(0) \right) \tag{2.35}$$

$$F_B(x) = \sum_{l=0}^{B} f_l^{(B)} x^l . \tag{2.36}$$

We determine $f_l^{(B)}$ by enforcing

$$\int_0^\infty dr\, r^{2m+1} \left[\bar{G}_B(r) - G(r) \right] = 0 \quad ; \quad m = 0, \ldots, B \tag{2.37}$$

using

$$\int_0^\infty dr\, r^{2m} \mathrm{erfc}\,(\sqrt{\alpha} r) = \frac{1}{\alpha^{m+1/2}} \frac{(2m)!}{2^m \sqrt{\pi} (2m+1)!!} \tag{2.38}$$

$$\alpha^l \int_0^\infty dr\, r^{2m+2l+1} e^{-\alpha r^2} = \frac{(m+l)!}{2\alpha^{m+1}} \tag{2.39}$$

Then

$$\int_0^\infty dr r^{2m+1} \left[\bar{G}(r) - G(r) \right] = -\frac{1}{\alpha^{m+1/2}} \frac{(2m)!}{2^m \sqrt{\pi} (2m+1)!!} + \frac{1}{2\alpha^{m+1/2}} \sum_{l=0}^{B} (m+l)! f_l^{(B)} , \tag{2.40}$$

so that

$$\sum_{l=0}^{B} (m+l)! f_l^{(B)} = \frac{(2m)!}{2^{m-1} \sqrt{\pi} (2m+1)!!} \tag{2.41}$$

2.4.3 Regulated Kernels for the $3d$ Wave Equation

One could apply the analogous procedure to the kernel for the wave equation:

$$G(r) = \frac{e^{ikr}}{r}, \tag{2.42}$$

however the required moments are not as easy to compute, and would depend on the additional dimensionless parameter ka. A simple and successful alternative is to use the regulated kernel for the Laplace equation [here denoted as $\bar{g}(r)$] as follows:

$$\bar{G}(r) = \bar{g}(r)\cos(kr) + \frac{i\sin(kr)}{r}. \tag{2.43}$$

(One must note that the imaginary part of G is *already* regular and should not be altered.) This (and its analog in $2d$) is what is implemented in FastScat.

2.4.4 Use of Regulated Kernels

The basic choices are

r_0 the distance beyond which the bare kernel is used,

$\alpha = 1/a^2$ the high frequency cutoff parameter, and

B the order of the correcting polynomial.

These will depend (at least) on the size and shape of the patches, the maximum basis function order, and the digits of precision required p. The size and shape of a patch will be parameterized by l, a "minimum" length. A "maximum" length L will govern the choice of quadrature order. For a $2d$ patch, these are locally,

$$l(u) = L(u) = \sqrt{g(u)}, \tag{2.44}$$

where one might evaluate at several points on the surface and choose the smallest l and largest L. For $3d$,

$$L(u) = \frac{\sqrt{6}}{3}\sqrt{g_{11} + g_{22} - g_{12}} \tag{2.45}$$

$$l(u) = \frac{\sqrt{g}}{L(u)}. \tag{2.46}$$

The proportionality constants are determined by requiring that for an equilateral triangle, L = side length and l = altitude. It is a good guess that setting B equal to the basis function order is nearly optimal.

We choose r_0 and α by setting

$$\left|\frac{\bar{G}_\alpha(r_0) - G(r_0)}{\bar{G}_\alpha(0)}\right| \approx 10^{-p} \tag{2.47}$$

A good fit for $2d$ is $1 \leq p \leq 16$ and $0 \leq B \leq 12$ is

$$\alpha r_0^2 \;=\; 0.0508181\, B + 3.26799\, p\,. \tag{2.48}$$

A good fit for $3d$ is $1 \leq p \leq 16$ and $0 \leq B \leq 12$ is

$$\alpha r_0^2 \;=\; 3.1729\, p\ . \tag{2.49}$$

One should not use $p < 1$. All more complicated fits tried lead to incorrect behavior.

Chapter 3

The Fast Multipole Method for the Wave Equation

3.1 Introduction

At the beginning of this contract, no one really knew how to construct a fast multipole algorithm in more than two dimensions. Our proposal said:

> *Construct a 3d version of the algorithm.* Conceptually, this is a relatively straightforward procedure. However, it presents a number of technical difficulties, associated with the cumbersome nature of the 3d partial wave expansions and related "addition theorems".
>
> Indeed, the principal analytical tools of Ref. [Rok90] are the partial wave expansions and translation operators for the Helmholtz equation in two dimensions. It turns out that the latter have an analytically available diagonal form, and it is the ability to apply them rapidly that results in a fast algorithm. Furthermore, in two dimensions, the translation operators are convolutions, and are diagonalized by the discrete Fourier transform (DFT). While not necessary for the construction of a fast scheme, the latter fact significantly simplifies the analytical apparatus needed. This advantage is not available in three dimensions, where the DFT is replaced with the decomposition of functions on a sphere into a series of harmonic functions. The theory of such decompositions and their behavior under translation operators is not as satisfactory as that in two dimensions, and some serious analytical work will be necessary.

As it turned out, the observation of Coifman and Rokhlin that the diagonalization of the $2d$ addition theorem by a DFT was physically equivalent to the use of a far-field representation immediately dispersed all "technical difficulties" and made it obvious how to realize the FMM in any number of dimensions. After the presentation[CRW93b, CRW93c, CRW93a] of this view, the FMM was understood and implemented not only by us, but by other groups (Boeing and University of Illinois), with stunning speed. This may be the most important result of this contract effort.

Beyond the extension to $3d$, there have been many important developments in FMM theory over the past four years. Because these have been well documented in the literature, we use this chapter to simply outline them and cite the relevant publications.

3.2 Classic $\left(\mathcal{O}\left(N^{3/2}\right)\right)$ single-stage FMM

The original exposition of the Helmholtz FMM (in $2d$) was given in [Rok90]. As mentioned above, the hurdle to application in higher dimensions was the "cumbersome" way in which the general addition theorems were viewed. (Some appreciation of this may be attained by reference to [Che90, Che92, WC93].) The key step of translation operator diagonalization in $3d$ is explained in [Rok93]. A prescription accessible to pedestrians, with some physical interpretation, was published in [CRW93a]. Detailed extrapolations of the theory were given by Epton[ED95]. The mechanism of the Boeing implementation is described in [DY94].

3.3 Faster $\left(\mathcal{O}\left(N^{4/3}\right)\right)$ single-stage FMM

It came as somewhat of a surprise that the classic single-stage FMM can be significantly improved. Again, a physical view elucidates the mathematical apparatus; the action of a source group on a well separated field region should depend only on the far-field of the source in the directions of the observation points. This results in a further reduction in the density of the translation operator matrix representation. Our approach is outlined in [CRW94]. The method was discovered independently by Wagner and Chew[WC94]. An extreme version (where the interaction between a group pair is represented by a single far-field direction) is the fast but crude "far-field" approximation[LC95]. At the end of the contract, we recognized a deep simile between the translation operator of the $\mathcal{O}\left(N^{4/3}\right)$ FMM and the method of high-order regulated kernels discussed in the preceding Section 2.4. In fact, the $N^{4/3}$ translation operator in the large separation limit is exactly a high-order regulated delta function.

3.4 FMM for periodic structures

For computation of the fields of a periodic structure, Fourier transform techniques obviate the use of the FMM, as long as the unit cells are not large compared to a wavelength[KTW93]. The difficulty is that a straightforward precomputation of the matrix elements of the periodic kernel is extremely expensive because it involves the use of the Ewald summation method in the integrations. By using the Ewald method on the FMM translation operator[RW94], the precomputation can be vastly reduced.

3.5 FMM with overlapping basis-function domains

A technical difficulty arises when employing overlapping basis functions to insure continuity. This point was discussed in the annual report[HOS+95c]. When basis functions with forced continuity are abandoned (see Section 2.2), this ceases to be an issue.

3.6 Multilevel FMM

In spite of some claims to the contrary, Rokhlin's original paper[Rok90] described an implementation of a multilevel $2d$ FMM. The multilevel apparatus was sketched in [CRW93a]. The major difference in $3d$ is that the interpolation (and filtering) needed to aggregate source (and "disaggregate" field) groups can not be made fast in a trivial way. Unfortunately, slow interpolation and filtering spoils the complexity of the multilevel FMM. Our annual contract report[HST+93c] provided an accurate interpolation prescription, but no fast implementation prescription. This problem can again be solved by use of a high-order regulated kernel, or by use of FFTs and a $1d$ FMM. The importance of fast interpolation and filtering is discussed in [DY95]. Implementation results are described in [SC95].

3.7 Parallel FMM Implementation

A prototype parallel implementation (of the single-stage "classic" FMM) is described in [Sta95]. It exhibited good scalability for problems up to a substantial size.

Chapter 4

Implementation

4.1 Introduction

A primary motivation behind FastScat development was to create a *general* program capable of efficiently and accurately computing scattering and radiation from surfaces of arbitrary shape and size. In this program, we have implemented conventional solution techniques as well as new computational algorithms, such as the FMM. In addition, FastScat has been used as a testbed to demonstrate the effectiveness of various enhancements such as accurate surface models, high-order basis functions, and accurate quadrature rules.

To support this work, FastScat had to be written in such a way as to be highly modifiable and extensible, as well as reasonably efficient. FastScat was designed using an object-oriented methodology[Mey88], and was written primarily in C++. In the rest of this chapter we elaborate on some important aspects of this approach to the implementation, and describe the unique features of the resulting program.

4.2 Logical Structure

The first, and probably most important, step toward the successful implementation of FastScat was laying out the overall design of the program. Although we would first consider scattering in $2d$ using conventional techniques, program designers had to look forward to the later extension to $3d$ surfaces, and to the addition of new methods of discretization and solution. The resulting FastScat design is based on the key abstractions of the physics of scattering. By defining classes such as `Surface`, `Patch`, `BasisFunction`, `GalerkinDiscretization`, `Z_Matrix`, `FMM`, and `Current`, we created a direct mapping of the theory and algorithms onto the resulting computer code. This class organization along with language features such as inheritance, data encapsulation, polymorphism, and

dynamic binding allowed the key elements of the problem to be expressed and manipulated in a natural way.

A good overview of FastScat's design can be found in [HST+93a], included in Appendix A. The paper describes our implementation objectives and how these led us to adopt C++ and an object-oriented design. It also introduces an number of important FastScat classes, and discusses some of the costs and benefits of our approach. A more up-to-date description of the implementation of surfaces in FastScat can be found in [HST+94], also included in Appendix A.

4.3 Programming Efficiency

One of the major benefits of our use of an object-oriented approach, and of a language with features that support this approach, has been realized in programming efficiency. The most significant effects on programming efficiency were seen with the addition of new features and algorithms to FastScat. Although it took nearly six months of design and coding to first compute scalar scattering in $2d$, the speed with which were able to implement curvilinear surface modeling, higher-order basis functions, new iterative solvers, the FMM, and the extension to $3d$ electromagnetic scattering has been impressive. Key to these successes, we believe, is our object-oriented implementation in which classes containing *hidden data representations* interact only through well-defined interfaces.

According to Brooks[Bro75], "Representation is the essence of programming," and thus true advances in the efficiency of a computation are usually recognized through changes in the data representation. Unfortunately, using traditional programming techniques, the data representation often becomes a common thread throughout the entire program, and thus changes to this representation result in many and far-reaching code modifications. In contrast, such changes are not expensive in a well designed C++ program. Consider, for example, the FMM, whose sparse components require a fundamentally different data representation than do traditional dense techniques. Having implemented the dense solution methods in terms of a general class `SurfaceOperator`, the addition of the FMM in $2d$ required only the creation of a new class, derived from `SurfaceOperator`, which would maintain the new sparse representation. Likewise, although initially no one was sure how to apply the FMM in $3d$, when an algorithm was finally defined implementation in FastScat was quick and simple. Even unanticipated changes, like moving to the regulated kernel and patch-based basis functions, were readily implemented within the FastScat framework. Such a basic change in approach would likely have been a disaster in a more traditional code.

4.4 Runtime Efficiency

At the beginning of this contract, there was debate over what language would be used for FastScat implementation. FORTRAN seemed a natural choice because it was widely known in the scientific community, and because highly optimized FORTRAN compilers were available on almost every platform. The disadvantages of FORTRAN, however, included the lack of dynamic memory allocation, and the lack of features to support object-oriented programming. Although the emerging standard for FORTRAN-90 promised relief from these shortcomings, few compilers (and fewer development tools) were available at the time.

C++[ES90], on the other hand, had the basic features necessary (such as classes, inheritance, and dynamic binding) for an object-oriented implementation. Because it was first implemented as a translator into C, the language was portable and widely available on supercomputers. Both compilers and development tools were available and reasonably mature on most platforms. The major concern was that the additional features of a language such as C++ would result in runtime overhead that could not be tolerated in a numerically intensive code.

After careful consideration of this issue, C++ was selected as the primary implementation language for FastScat. Recognizing that most of the execution time of the program (written in any language) would be spent computing integrals and doing basic linear algebra, we took great care to insure that these *computational kernels* were as efficient as possible. For instance, linear algebra in FastScat's vector and matrix classes was implemented in terms of LAPACK[ABB+92] and the BLAS[DCDH90, DCHH88], highly optimized FORTRAN libraries individually tuned on most machines (including several parallel and vector machines). Using these FORTRAN functions, called from C++, we were able to keep all the benefits of object-oriented programming with minimal sacrifices in performance.

The sacrifices in efficiency we did encounter in our C++ implementation were often due to the relative immaturity of C++ compilers rather than fundamental limitations of the language itself. Most have disappeared with the next-generation compilers now being used. For example, virtual function calls, used extensively by the surface and basis function classes, were initially very slow and caused some code rewrites in which virtual functions calls were removed from within large loops. This problem, however, vanished with later compiler versions, as did other performance bottlenecks. Overall, we believe that the minimal (and in some cases, short lived) performance penalties associated with C++ are overshadowed by the flexibility gained by utilizing its object-oriented features. We expect similar results could be obtained in other scientific programs by judicious use of optimized computational kernels embedded within a flexible, object-oriented code.

4.5 Features and Abilities

Currently, FastScat computes scattering from bodies represented in two or three dimensions. Applying a method of moments formulation, it can compute cross sections with Dirichlet or Neumann boundary conditions. In $3d$, it can compute electromagnetic scattering from perfect conductors. FastScat has a host of features which make it unique among scattering codes. Some are described here.

Accurate Surface Models Surface models in FastScat are composed of triangular patches which may be flat or curved. A sphere, for instance, can be patched with flat facets, or with patches which exactly conform to the surface. Use of curved patches can eliminate artificial creases or edges in the surface model, providing a more accurate representation of the scattering surface.

Adjustable Accuracy of Computation FastScat has the ability to use high-order quadrature and careful singularity removal to preserve the accuracy and efficiency of its calculations. Program accuracy, however, is adjustable and can be set by the user.

High-Order Basis Functions FastScat allows the use of high-order basis functions to model current on the surface. With an accurate surface model and collections of high-order basis functions, one can adequately describe the surface current using fewer patches and unknowns than allowed with low-order functions. This permits a significant reduction in the CPU time and storage required to achieve a given accuracy in modeling a scatterer or antenna[HRS+93]. Because the cost of obtaining even greater accuracy is relatively low, by increasing the basis function order in a given problem, the user can easily estimate the accuracy of a solution.

Solution Methods The user can select one of three solution methods in FastScat: direct LU decomposition, a dense iterative technique, and the FMM. With an iterative solution method, three iterative solvers are available: conjugate gradient, biconjugate gradient, and bistatic biconjugate gradient.

Chapter 5

Results

5.1 Validation

5.1.1 Introduction

To validate an RCS code is to verify that it computes cross sections accurately. Sources of inaccuracy (i.e. differences between a calculated RCS and the actual RCS for a given scatterer) can be classified into two broad categories —

- deficiencies in the abstract model of the scatterer and
- deficiencies in the way the RCS for the abstract model is calculated.

The former category is modeling error. It includes such simplifications as modeling a surface as a perfect conductor, treating thin body parts as though they actually had zero thickness, etc. The latter category is solution error. When two codes agree on an abstract model, differences in their calculated RCS's are a reflection of the different ways they arrive at approximate numerical solutions to Maxwell's equations for the boundary conditions appropriate to that model of the scatterer.

With method of moments RCS codes it is possible, at least in principle, to reduce solution error to a negligibly small value by reducing the discretization scale size. A principal design goal of FastScat is to also make it practical. FastScat attempts to accomplish this goal by employing high-order methods. When the solution error has been rendered negligible, the focus can shift to improving the correspondence between the abstract model and the actual scatterer in order to achieve better agreement between calculation and experiment.

In this section we discuss different ways to assess the accuracy of a computed RCS, how accurately FastScat computes the RCS's for various two- and three-dimensional scatterers, and the effectiveness of FastScat's high-order methods (quadratures, current basis functions, and surface representation) in facilitating

rapid convergence to the correct answer. A more detailed discussion of this topic can be found in [HOS+95b] and [HOS+95a].

5.1.2 Accuracy

The first step in the validation of an RCS code is computation of cross sections for geometries whose exact cross section is known *a priori*. In this case it is a trivial matter to determine the accuracy of the computed RCS. Unfortunately there are only a few types of scatterers, such as the circle in $2d$ and the sphere in $3d$, that fall into this category. For both geometries there are Mie series solutions from which arbitrarily accurate cross sections can be easily computed. Both geometries also happen to have the kind of smooth surfaces for which the high-order methods incorporated into FastScat exhibit their maximum benefit.

For the vast majority of problems the exact answer is, of course, not known ahead of time so it is impossible to measure the error of an RCS calculation directly. Nonetheless, it is possible to estimate the accuracy of a computed RCS (for a given abstract model) by reliance on the assumption that the solution will converge to the correct answer as the discretization scale size decreases. If, in a series of RCS calculations for a given problem, the solution converges as the discretization becomes finer and finer, then one can conservatively estimate the error in the solution with the finest discretization to be of the order of the difference between this solution and that of the next finest discretization. While it is possible for the series of computed solutions to converge to a wrong answer, if the code employs high-order methods and the observed rate of convergence is high-order, this is very unlikely.

We have also compared FastScat's results to those of other RCS codes and to experimental data for purposes of validation. For example, we compared FastScat's results to those of RAM2D and CARLOS-3D for identical abstract models of scatterers. In all cases, the results agreed, although the precision of agreement was limited by the solution accuracy of the comparison code. The "business card" is a problem for which we compared the results of a FastScat computation to experimental data. A detailed description of this scatterer and a discussion of the RCS comparison can be found in Section 5.1.2. We observed noticeable differences that can be attributed partly to modeling error and partly to measurement error.

Large Dynamic Range

We are particularly interested in computing scattering from objects which have a large dynamic range in their cross sections. These types of computations put greater stress on the numerical algorithms than computations in which the cross section does not vary dramatically as a function of angle. Under these circumstances, the advantages of using high-order methods to achieve acceptable solution accuracy at all viewing angles become dramatically evident.

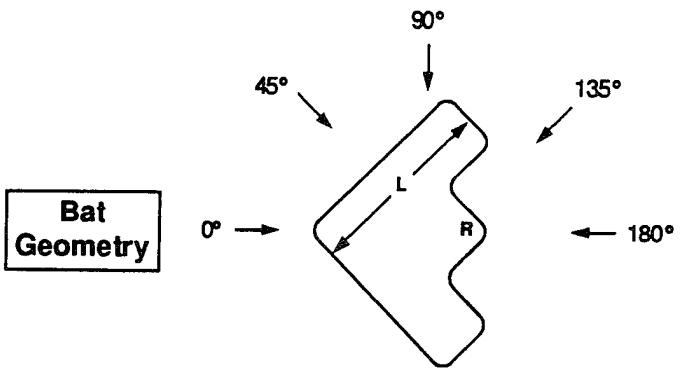

Figure 5.1: Geometry and incident angle definitions for the $2d$ "bat".

Discretization studies An example of a $2d$ geometry with a large dynamic range in the RCS is the "bat", shown in Figure 5.1. It is composed of straight faces connected smoothly by circular arcs of radius R. There are two long edges of length L and six short edges, each of length $L/3$, at right angles to each other. All surfaces are perfect conductors. It has three high-RCS specular reflection regions (one of which is the $2d$ analog of a corner cube) and a low RCS everywhere else. With the dimensions of the bat arbitrarily chosen to be $R = 1\lambda$ and $L = 300\lambda$, we computed the monostatic RCS for TM scattering using high-order current basis functions and using low-order current basis functions. The number of unknowns was fixed at 6000 for both calculations so that accuracy could be compared for comparable computation costs. In both cases, an exact surface model was used and the quadrature order was set high enough so that the additional error due to inaccurate quadratures was negligible.

The results are shown in Figure 5.2. The upper plot shows the result of discretizing the current on the bat by means of fourth-order CBF's and one wavelength long patches. There are narrow peaks at 45° and 135° as expected and a broader peak centered at 180° resulting from the "corner square" effect. The oscillations evident in the cross section are the result of interference, not due to any solution error. By comparing it to the solution computed with fifth-order Current Basis Functions (CBF) we have determined that it is accurate to better than two significant digits at all angles. In other words, the error is much less than the width of the plotted line. The result of using zeroth-order CBF's and one-fifth wavelength patches is plotted below. Whereas this calculation generally agrees with the high-order calculation at angles where the cross section is high, in the low cross section regions the calculated RCS is clearly erroneous, the error exceeding 20 dB at some angles.

Figure 5.3 demonstrates this effect for scattering from a larger bat with a similar geometry.

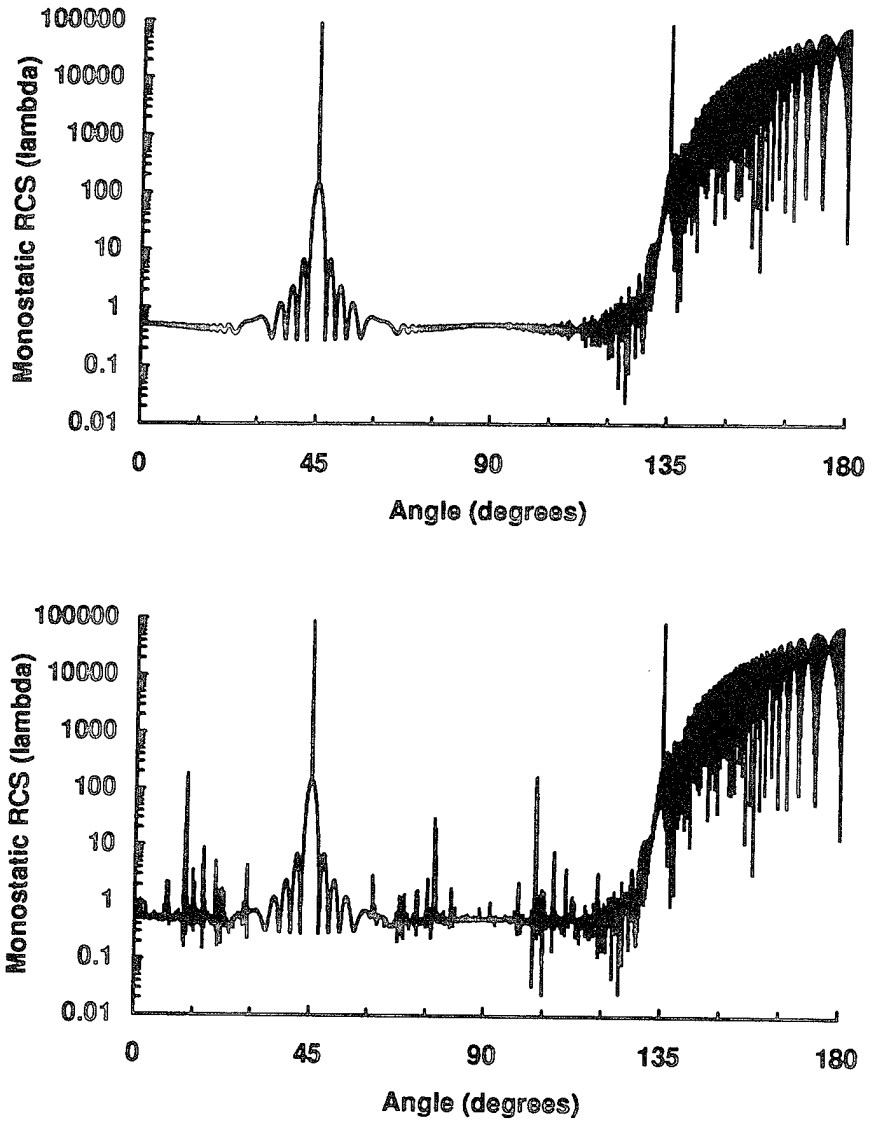

Figure 5.2: Monostatic TM RCS for "bat" computed by FastScat using 6000 unknowns. Upper plot: fourth-order current basis functions & 1200 patches. Lower plot: zeroth-order (i.e. pulse) current basis functions and 6000 patches.

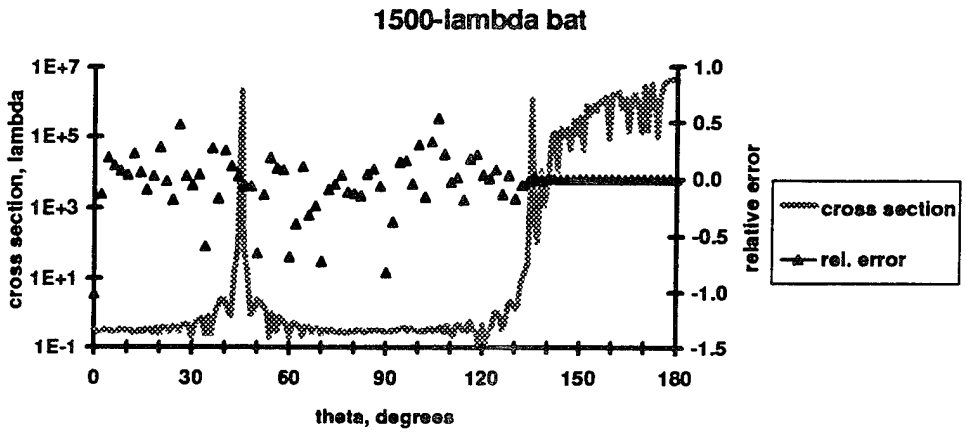

Figure 5.3: Cross section error computing scattering from a 1500λ bat with Dirichlet boundary conditions.

Quadrature order studies There is an interesting, incompletely understood sidelight to this problem having to do with the quadrature accuracy required to achieve a certain RCS accuracy. Suppose we consider separately the far-field radiation patterns of each patch comprising the discretized bat. Then we can view the regions of high cross section to be the result of mostly constructive interference of the far fields and the regions of low cross section to be the result of near cancellations of the far fields due to destructive interference. Given this interpretation, we might expect that achieving one digit of accuracy in the low cross section regions (where the RCS is about five orders of magnitude lower than its peak value), should require that each element of the impedance matrix be accurate to at least six digits. For the bat, as well as some other scatterers we have investigated, this turns out not to be the case. In fact, we determined the cross sections to be accurate to better than one digit at all angles even when only four quadrature digits were requested (i.e. when the quadrature order given by an empirical formula was set to produce impedance matrix and excitation vector elements accurate to about four digits). Although we have not yet resolved this puzzle conclusively, we suspect the answer lies in the high degree of symmetry and uniformity among the patches, which could well result in very precise cancellations between far fields that are individually less accurate.

Series comparisons With high-order quadratures, current basis functions, and surface representations in its arsenal, FastScat can calculate very accurate cross sections with fewer unknowns and in less time than codes restricted to using low order methods. In subsequent sections we demonstrate how each high-

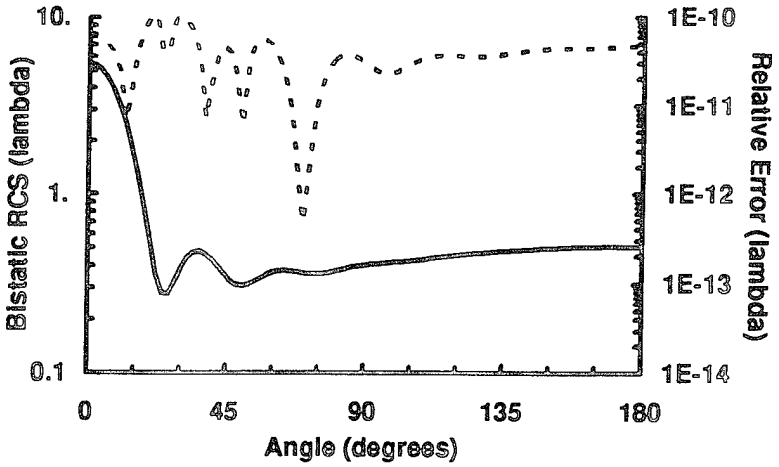

Figure 5.4: Solid line / left ordinate — RCS of 1λ-radius circle for scalar scattering in the TM polarization as a function of bistatic observation angle. Dashed line / right ordinate — Relative error between Mie series solution and FastScat result.

order method contributes to the overall high-order convergence of the computed RCS to the exact answer. However, before doing so it is useful to show the accuracy to which cross sections computed by FastScat can be validated on a few sample geometries. For this comparison we have chosen two geometries for which exact answers are known (circle and sphere), several for which the solution accuracy can be estimated by observing solution convergence (eg. ellipse, ogive and several airfoils), and one for which experimental data is available ("business card").

The surfaces of the circle in $2d$ and sphere in $3d$ are smooth, resulting in smoothly-varying surface currents that can be approximated rather efficiently using our sets of high-order CBF's. The cross section for (scalar) scattering from a 1λ-radius circle in the TM polarization is plotted as a function of bistatic observation angle in Figure 5.4. On the same graph we have plotted the relative difference between the Mie series solution and the result computed by FastScat using 8 equal-sized patches, tenth-order CBF's, and an exact surface representation. Figure 5.5 shows analogous results for bistatic (vector) scattering from a 1λ-radius sphere with $\theta\theta$ polarization. In this case, the FastScat computation used fifth-order CBF's and an exactly-represented sphere that was subdivided into 180 nearly-equal-area patches. As the figures show, the FastScat results are accurate to at least 10 and 6 digits, respectively, at all observation angles. For small circles and spheres, solution accuracies approaching machine precision have been demonstrated by employing even higher order CBF's or finer

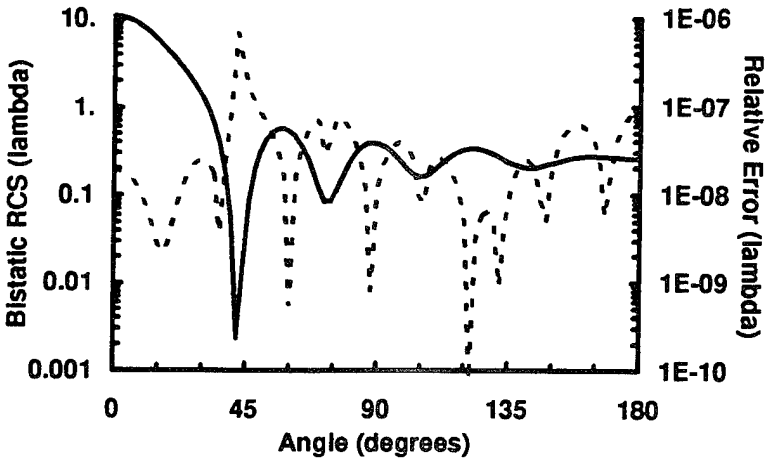

Figure 5.5: Solid line / left ordinate — RCS of 1λ-radius sphere for vector scattering with $\theta\theta$ polarization as a function of bistatic observation angle. Dashed line / right ordinate — Relative error between Mie series solution and FastScat result.

patching.

Other shapes We have also estimated our accuracy for scattering computations from shapes where no closed form solutions for the cross sections were available. For example, we have included a report in the appendix of some computations we did for the Second International Conference and Workshop on Approximations & Numerical Methods for the Solution of the Maxwell Equations in 1993, hereafter referred to as the GWU conference. For this workshop we computed scattering from ellipses, an ogive, single and double airfoils, and cavities.

We estimated the error in each solution by comparing our computations with a given discretization with those using a finer discretization. Because of our use of higher order methods, this type of error estimate was not very expensive. For example, on the smallest ellipse problem it took us 7 seconds to compute the cross section to one significant digit and 10 seconds to compute an answer accurate to 2 digits. Had we used low order techniques, it would have taken 20 to 50 times as long to get the more accurate answer.

Other participants at the workshop computed scattering cross sections from these same shapes using different techniques. Since our computations were the only ones with reliable error estimates, they were used as the standard by which the accuracy of other computations was judged. Our run times were significantly faster than those of the other participants. On workstations we computed cross

sections in less than a minute which were more accurate than those computed by others requiring hours of supercomputer time.

Data comparison We have also checked the accuracy of our computations by comparing our results to benchmark measurements made by the Electromagnetic Code Consortium. Figure 5.6 compares FastScat's computation of the cross section for scattering from a thin rectangular plate (the EMCC "business card") with measurements made at China Lake[WSWS92]. The FastScat results are identical to those computed by others. The differences between these computations and measurement seem to be due to measurement uncertainties including such factors as registration error, non-zero plate thickness, finite solid angles in the transmit and receive horns, and the fact that these measurements were made in the near field.

5.1.3 High order quadratures

We have verified that our $2d$ and $3d$ quadrature techniques are high order. This was done by computing scattering from a circle and a sphere using a combination of patch size and basis function order sufficiently small to eliminate all sources of error besides quadrature errors.

Unregulated Kernel

We have used two quadrature techniques in FastScat, both of which are high order. The first, more traditional, approach does numerical integrations using Gaussian-type quadrature rules. Singularities in the integrands are dealt with by changes of variable which eliminate them or isolate them to end points of the integration. Endpoint singularities are integrated using special quadratures rules derived for this purpose.

The quadratures are considered to be high order if the achieved accuracy grows rapidly with the time taken to perform the numerical integrations. Two demonstrations that these quadratures using the unregulated kernel are high-order are presented below. The first is a $2d$ problem scattering from a circle. The second is for $3d$ scattering from a sphere.

Circle We checked the quadrature accuracy for scattering from a 5λ radius circle by making all other sources of error negligible. To find an appropriate discretization, we divided an exact circle into 60 equal patches. We computed the monostatic scattering cross section for a Dirichlet boundary condition with exceptionally high quadrature order for various numbers of basis functions on each patch. These results were compared to a Mie series solution good to 13 digits, $d\sigma/d\theta = 2.501493628335\lambda$.

Table 5.1 and Figure 5.7 show the error in the cross section as a function of the order of basis functions used on each patch. As can be seen in the figure

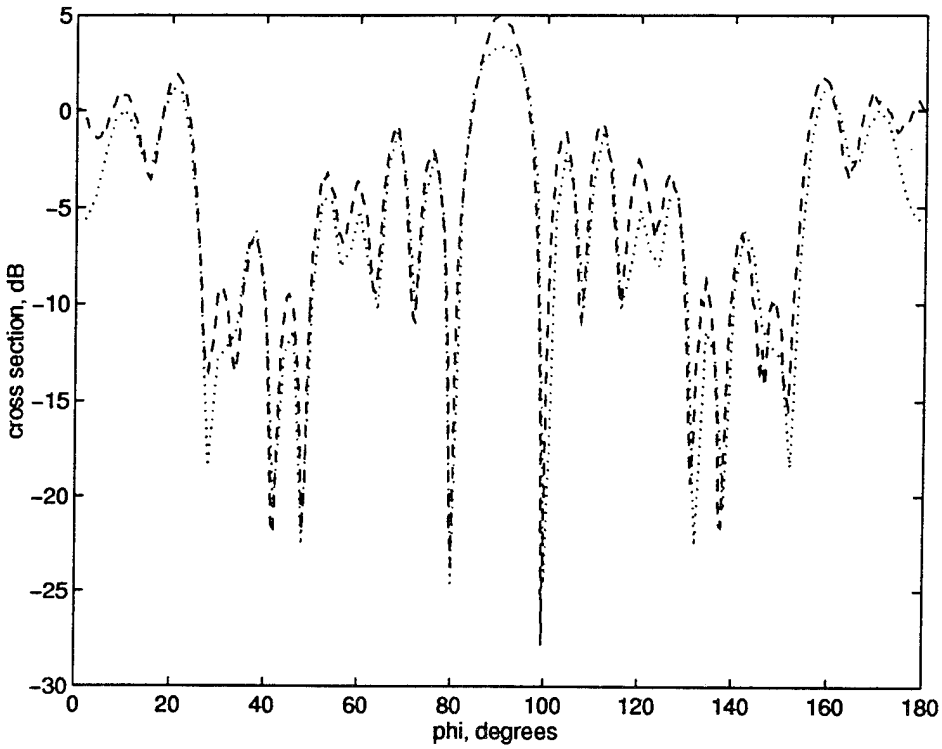

Figure 5.6: Cross section for scattering from a 3.5λ by 2λ plate with an elevation angle of 10 degrees. The polarization is parallel to the edge of the plate. The dashed curve is the measured data and the dotted curve the cross section computed by FastScat.

bfo	error
0	2.1
1	9.5e-3
2	7.1e-4
3	2.0e-5
4	4.0e-7
5	4.5e-9
6	5.6e-11

Table 5.1: Error in the monostatic cross section from a 5λ circle as a function of basis function order on each patch.

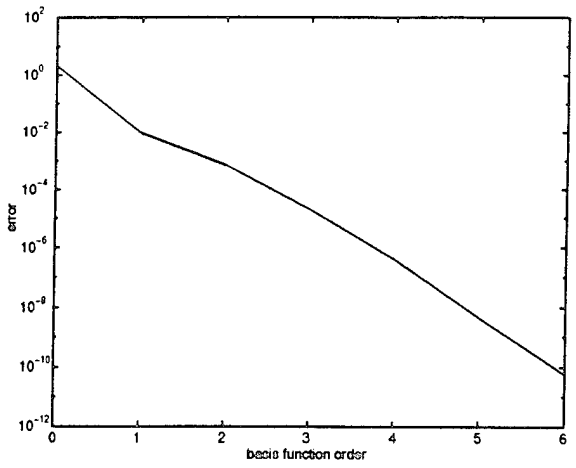

Figure 5.7: Error in the monostatic cross section from a 5λ circle as a function of basis function order on each patch.

and table, the cross section converges in a high order manner to have an error less than 10^{-10} for basis function order 6.

Using the above discretization with basis function order 6, the cross section was recomputed with various accuracies for the quadratures. The results are shown in Table 5.2 and Figure 5.8. The fill times shown are CPU seconds on a Sun SparcStation 10. There are 420 unknowns. The convergence of the quadrature accuracy with CPU time for this problem is exceptional. By increasing the matrix fill time by 64% the error of the quadratures as reflected in the cross sections can be lowered by eight orders of magnitude.

Sphere A similar study to that reported above for a circle was done with a 0.5λ radius sphere. First, the monostatic cross section was computed with very accurate quadratures (6 digits were requested) for various basis function orders to establish what discretization was necessary to eliminate all but quadrature errors. Table 5.3 and Figure 5.9 show the results of this study. The relative error as computed using the exact Mie series result of $d\sigma/d\Omega = 0.04727522254319971$.

Using the same discretization with basis function order 7, the cross section was recomputed with various accuracies for the quadratures. The results are shown in Table 5.4.

digits	fill	error
0	9.1	2.0e-2
1	10.3	6.0e-2
2	10.8	2.8e-5
3	12.2	2.4e-5
4	12.9	3.8e-8
5	14.6	3.7e-8
6	14.9	2.4e-10

Table 5.2: Cross section error for monostatic scattering from a 5λ radius circle with Dirichlet boundary conditions. The circle had 60 equal patches and sixth order basis functions on each patch. The "digits" column is the number of digits requested in FastScat. The fill time is the total number CPU seconds to compute all matrix elements on a Sun SparcStation 10.

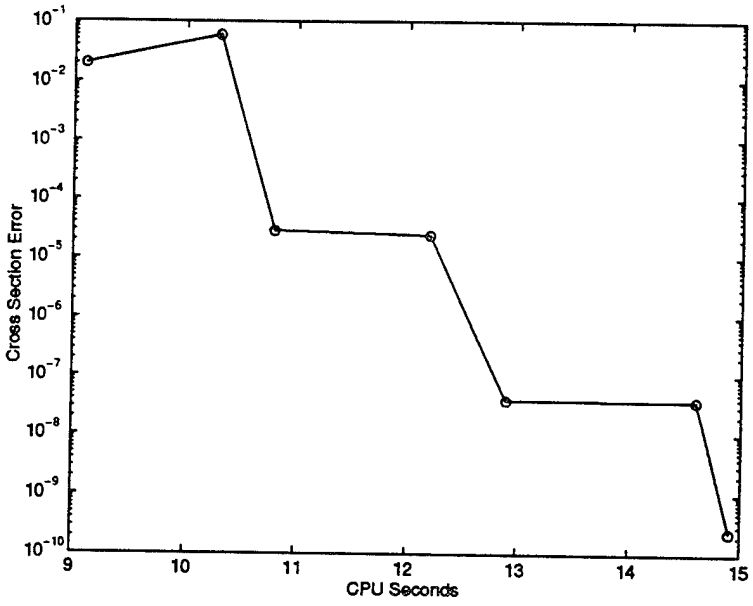

Figure 5.8: Cross section error for monostatic scattering from a 5λ radius circle with Dirichlet boundary conditions as a function of matrix fill time. The circle had 60 equal patches and 6th order basis functions on each patch. The fill time is the total number CPU seconds to compute all matrix elements on a Sun SparcStation 10.

order	error
0	4.1
1	1.2e-1
2	1.8e-2
3	4.6e-3
4	1.4e-3
5	1.1e-4
6	3.7e-5
7	1.4e-6

Table 5.3: Relative cross section error as a function of basis function order for monostatic scattering from a 0.5λ sphere. Quadratures were done accurately enough for at least 6 digits of precision.

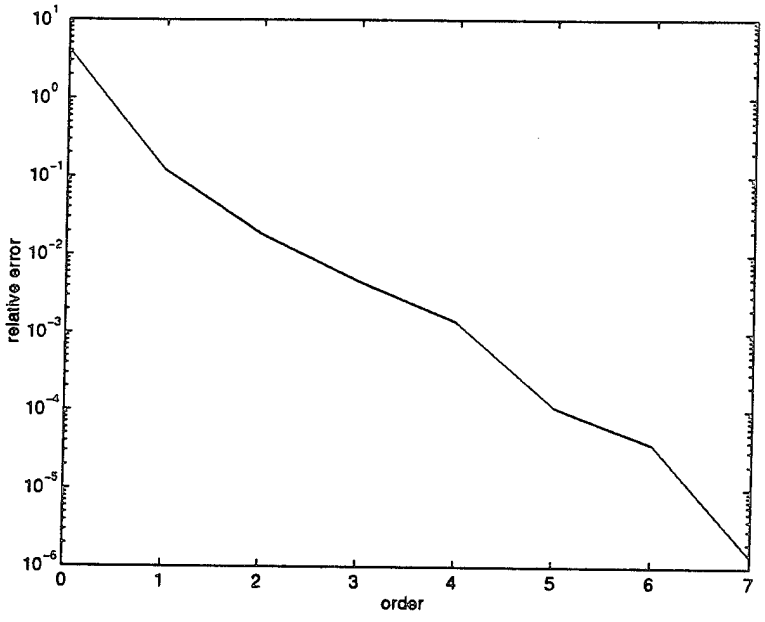

Figure 5.9: Cross section error for monostatic electromagnetic scattering from a 0.5λ radius sphere as a function of basis function order. The sphere had 20 equal patches and a sufficiently high quadrature order to get about 6 significant digits in the impedance matrix elements.

digits	fill	error
0	71828	5.4e3
1	73388	.31
2	79916	2.5e-4
3	81487	2.0e-6

Table 5.4: Cross section error for monostatic electromagnetic scattering from a 0.5λ radius sphere. The sphere had 20 equal patches and seventh order basis functions on each patch. The "digits" column is the number of digits requested in FastScat. The fill time is the number CPU seconds to compute all matrix elements on a Sun SparcStation 10.

Regulated Kernel

The quadratures done in FastScat with the regulated kernel are also high order. Establishing this is less direct than is possible with the unregulated kernel. This is because the current discretization and kernel regulation are linked. The current discretization error can not be made negligible independent of the quadrature error (as one can do with the unregulated kernel). However, having already established that the current discretization is high order, one can examine the convergence of the combined current and quadrature discretizations. If the result is high order, then both the current discretization and the quadratures must be high order.

We have computed scattering from a 1λ radius sphere using two programs, FastScat and CARLOS-3D[PMMG92]. CARLOS-3D is a benchmark Method of Moments program distributed by the Electromagnetic Code Consortium which uses low-order methods. The results of the two calculations are given in Tables 5.5 and 5.6. The maximum cross section error as a function of the matrix fill time is compared in Figure 5.10. The RMS relative errors are compared in a similar way in Figure 5.11. For low accuracy, the FastScat quadratures are a little faster than those in CARLOS-3D. As the accuracy increases, the high-order methods in FastScat make it increasingly attractive over a low-order code.

5.1.4 High order basis functions

We have demonstrated the high-order convergence of our $2d$ and $3d$ scalar and vector basis functions. Some of these results were presented in IEEE/AP-S symposia in 1993[HST+93b] and 1994[HMS+94a]. We will discuss several additional cases below.

High-order methods reduce the power by which the cost grows as a function of accuracy for fixed problem size. Specifically, we expect the cost to scale as $(1/\epsilon)^{\alpha/\rho}$, where ϵ is the error and the value of α depends on the cost being

unknowns	fill time	RMS relative error	RMS max error
30	0.18	1.2	105
120	1.6	0.12	4.6
270	4.7	0.10	9.0
480	13.3	0.042	3.6
750	31	0.023	1.9
1080	64	0.014	1.2
1470	117	0.0096	0.79
1920	203	0.0071	0.57
2430	324	0.0053	0.43
3000	491	0.0042	0.33

Table 5.5: Error in the scattering cross section of a 1λ radius sphere computed using the CARLOS-3D program.

unknowns	fill time	solve time	RMS rel. error	max. rel. error
12	0.35	0.01	0.10779	3.32121
120	9.94	0.26	0.06164	2.47027
210	23.48	1.17	0.01812	0.92412
340	63.89	4.56	0.01440	0.89589
210	108.64	1.19	0.01053	0.25403
340	236.32	4.56	0.00324	0.06576
510	448.96	14.69	0.00091	0.05558
840	547.44	72.16	0.00061	0.04110
1360	1196.52	305.14	9.8e-05	0.00789
2040	2427.18	999.15	1.5e-05	0.00122
1590	3749.14	472.31	5.7e-06	0.00019

Table 5.6: Error in the scattering cross section of a 1λ radius sphere computed using the FastScat program.

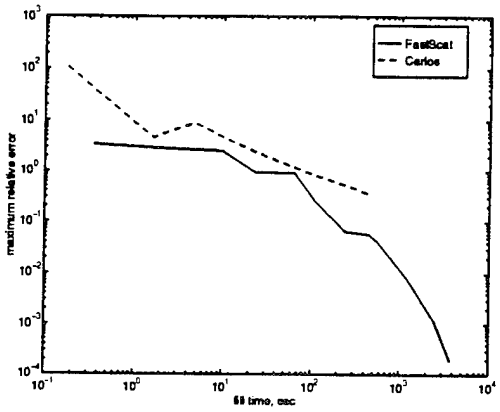

Figure 5.10: Comparison of the accuracy of CARLOS-3D and FastScat as a function of matrix fill time. The error shown is the maximum error in the bistatic cross section for scattering from a 1λ radius sphere.

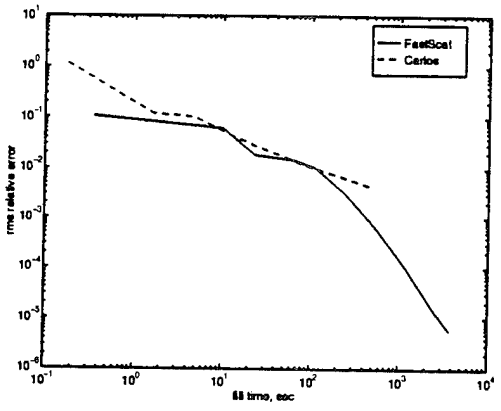

Figure 5.11: Comparison of the accuracy of CARLOS-3D and FastScat as a function of matrix fill time. The error in each case is the RMS relative error in the bistatic cross section for scattering from a 1λ radius sphere.

measured (e.g. α is 2 for memory and 3 for CPU time for a dense/direct solver) and ρ is the order of the numerical method. Therefore, on a log-log plot of error vs. cost, method order is revealed by the slope of the line connecting points corresponding to a given method order, but different sized patches. For several of the plots that follow, in which the cost measure is memory, we have chosen for convenience to plot the unknowns, N, rather than memory (which scales as N^2), on the abscissa. The error parameter plotted on the ordinate is either maximum relative error or RMS error. Both of these are useful measures for consolidating into a single number the error vs. observation angle data associated with each point.

$2d$ bodies

We have examined the high-order convergence of the current basis functions for several geometries. In this section we will present data for a 5λ circle, a wedge-circle, a rounded square, and a bat-shaped object.

5λ radius circle We studied the convergence of the cross section to the exact answer for bistatic scattering from a 5λ circle with a Dirichlet boundary condition. An exact circular surface and sufficiently high order quadratures were used so that all errors besides discretization error were negligible. The accuracy was measured by comparing with the exact Mie series solution for the monostatic cross section, $d\sigma/d\theta = 2.501493628335\lambda$. The results are shown in Figure 5.12. As expected, the slope of the curves decreases with increasing basis function order. The utility of using high-order basis functions is demonstrated by the observation that the number of unknowns required to achieve a given accuracy generally decreases as the basis function order increases.

Wedge-circle We also checked for convergence using the 6λ-long wedge-circle shown in Figure 5.13. The RMS error in the cross section was computed as a function of the number of unknowns in the problem using various orders of basis functions and various patch sizes. As is seen in Figure 5.14, first order basis function converge better than zeroth-order ones. Using basis functions of order higher than the first do not seem to help much for this problem. This is probably due to the geometrical singularity where the curved pieces join the straight ones. Tapering the patch size near these junctions would probably restore the expected high-order behavior of these basis functions.

Rounded square A $2d$ example problem that clearly demonstrates the relationship between current basis function order and rate of convergence of the final RCS is illustrated in Figure 5.15. The TM and TE bistatic cross sections for this body (verified to be accurate to better than 10 significant figures) are plotted in Figure 5.16. The rounded square is simply a square whose corners

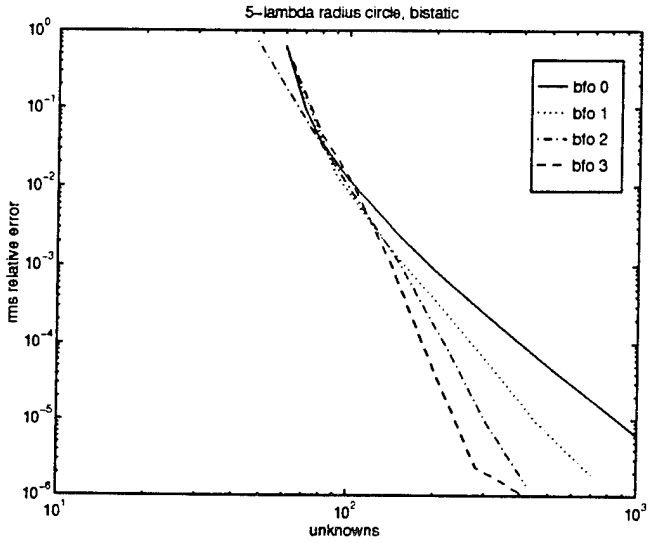

Figure 5.12: RMS relative error in the cross section for Dirichlet scattering by a 5λ-radius circle using various order basis functions and patch sizes.

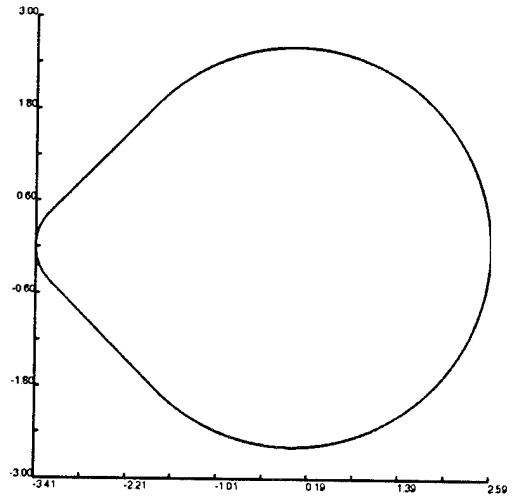

Figure 5.13: Wedge-circle geometry used for scattering calculations.

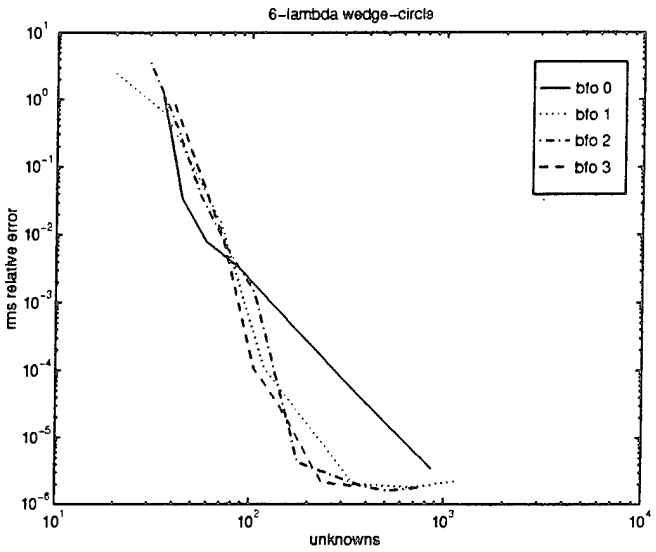

Figure 5.14: Relative RMS error in the scattering cross section from a 6λ-long wedge circle with a Dirichlet boundary condition. Each curve is for a different current basis function order.

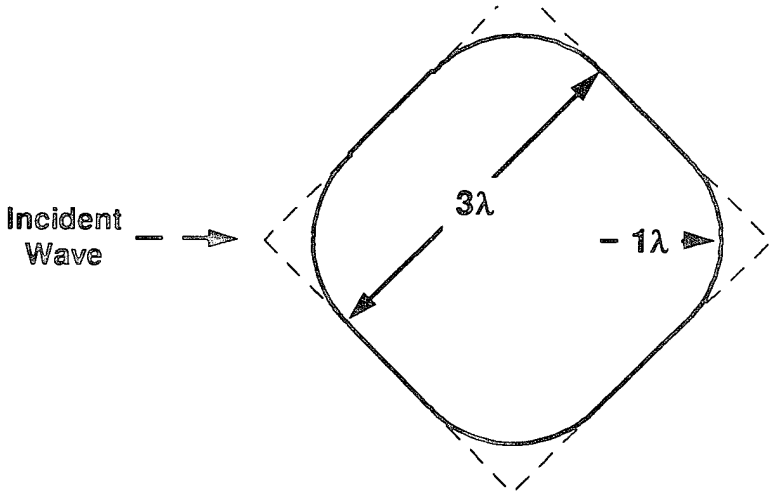

Figure 5.15: Rounded-square geometry inscribed inside a square (dashed line). Bistatic observation angles are measured relative to the incident wave direction.

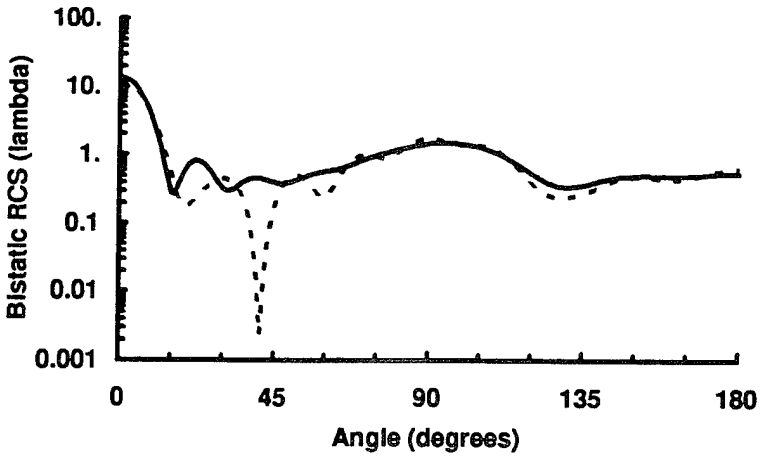

Figure 5.16: Plot of the bistatic cross section of a $R = 1\lambda$, $L = 3\lambda$ rounded-square from a FastScat calculation accurate to more than 10 significant figures at all angles. Solid line — TM polarization; Dashed line — TE polarization.

have been rounded into circular arcs. The arcs make the surface smoother so that high-order convergence is possible with our present set of 2D current basis functions without our having to resort to patch-size tapering at sharp corners. In FastScat's computations of the bistatic TE cross section, an exact surface representation was used and the quadrature error was made negligible, so discretization error is the dominant source of error. Figure 5.17 is the plot of maximum relative error vs. unknowns for this scatterer. Lines connect data points of the same current basis function order and different numbers of patches. The patches, approximately equal in size, varied in number by powers of two. The slopes of the lines increase as the method order increases. This is conclusive evidence that the use of higher-order current basis functions results in higher-order convergence of the RCS.

It is interesting to note how the lines associated with CBF-orders three and four start out steeper than the order two line, but eventually bend over and become parallel to it. Even though the tangent to the surface is continuous everywhere, higher derivatives of the surface are discontinuous (or worse) at each point where a circular arc meets a straight segment. In the mathematical sense, the surface has a singularity at these points. This forces the surface current to be singular at these points as well, eventually breaking the solution convergence rate for current basis function orders higher than two.

Bat The final $2d$ geometry for which we present basis function order data is the bat-shaped object shown in Figure 5.18. The cross section for monostatic

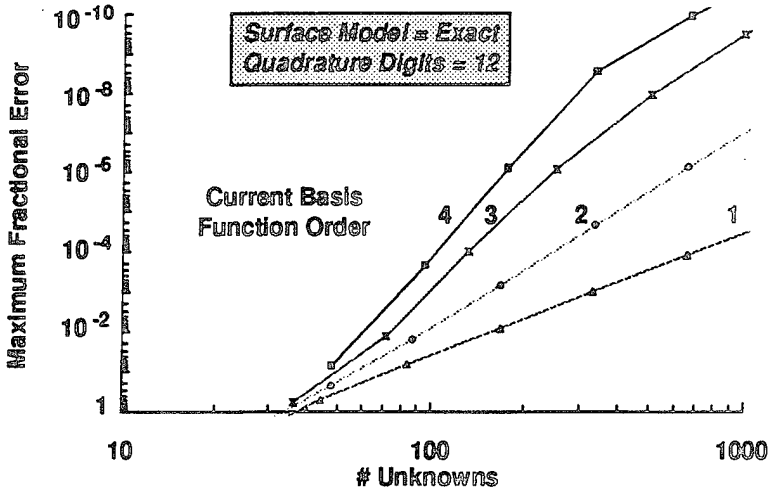

Figure 5.17: FastScat convergence rate on the rounded-square geometry as parameterized by current basis function order. The rate of convergence to the correct answer increases with increasing current basis function order. The "anomalous" behavior of order 3 and 4 curves at high accuracies is explained in the text.

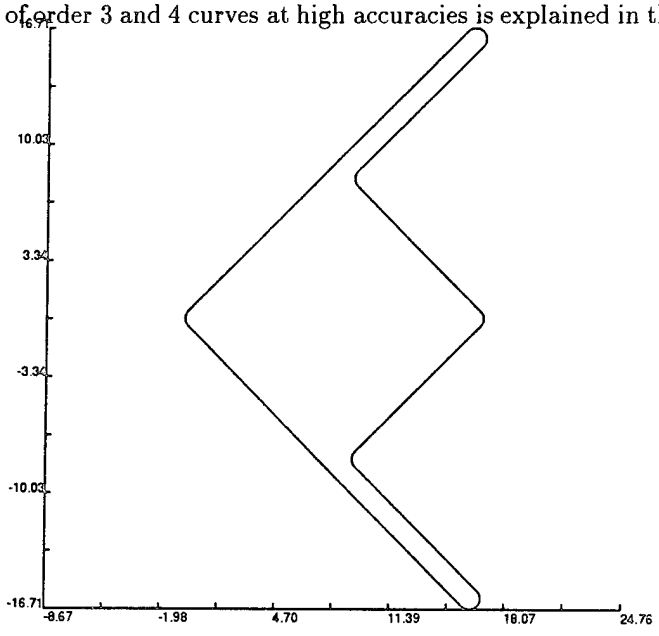

Figure 5.18: Geometry for 24λ bat used in the basis function order studies.

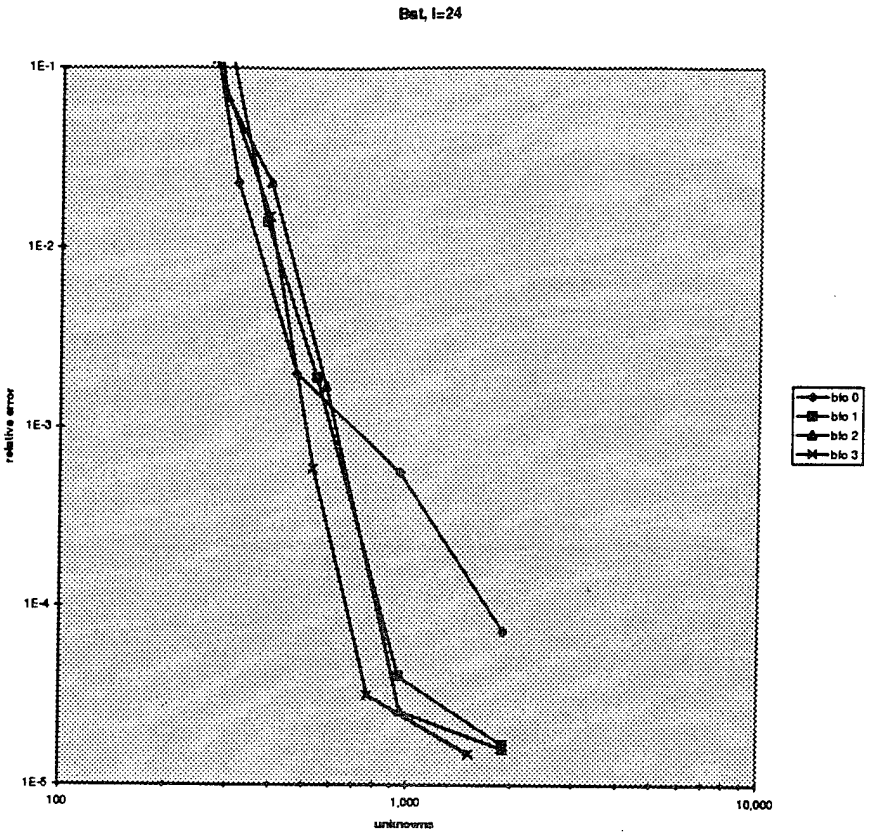

Figure 5.19: RMS relative error in the cross section for monostatic scattering from a 24λ bat as a function of the number of unknowns. Each curve is for a different basis function order.

scattering with a Dirichlet boundary condition was computed for incident angles from 0 degrees (nose on) to 90 degrees, in one degree increments. Figure 5.19 is a graph of the cross section error as a function of the number of unknowns for various basis function orders. The advantages of using high-order basis functions are not as evident in this figure as with other geometries, although the higher-order basis functions still show some advantage in the number of unknowns for high-accuracy solutions. We believe greater benefits are not seen because of the singularities in the higher derivatives of the slope of the unit surface normal where the straight sections of the bat join the semi-circles. Another benefit, not obvious from the graph is the reduced time to compute impedance matrix elements for a given number of unknowns when high-order basis functions are used. Because many basis functions are evaluated at the same points on each patch, implementation optimizations can be made which significantly reduce the quadrature times. For problems of this size, the computation of these matrix

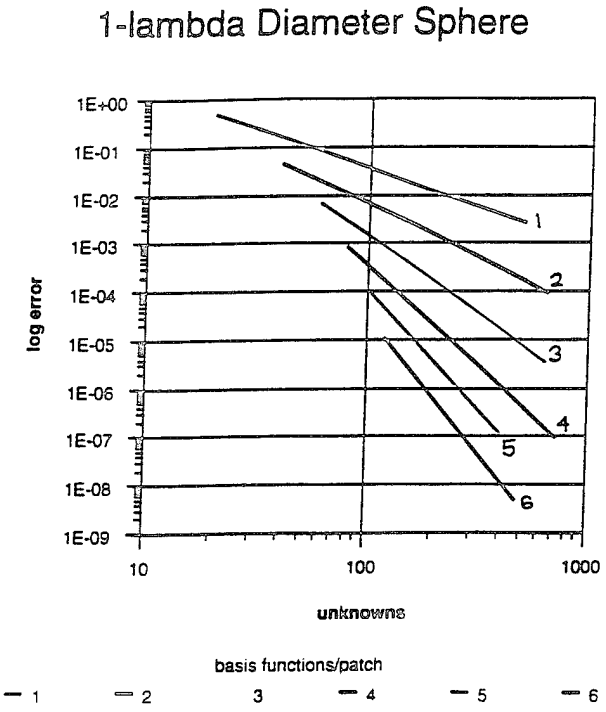

Figure 5.20: Error in the scattering cross section from a 1λ sphere versus number of unknowns for basis function orders $1 \leq o \leq 6$.

elements dominates the total CPU time.

$3d$ bodies

Sphere A good demonstration of the utility of high-order basis functions for $3d$ problems is shown in Figure 5.20. The solution accuracy for the RCS from a 1λ sphere versus number of unknowns is plotted for various basis function orders. The fact that our $3d$ basis functions are indeed high-order is clearly demonstrated by the shape of these curves. The slope becomes more negative with increasing basis function order, consequently the error scales as a power of some characteristic patch length l: error $\propto l^{-\beta o}$, where o is the basis function order and β is a constant. Recall that for fixed problem size and fixed basis function order, the number of unknowns is proportional to the number of patches.

5.1.5 Exact surfaces

As discussed in the 1993 IEEE AP-S Symposium[HRS+93], it is important to use surface models which accurately represent the CAD, mathematical, or manufactured surface. Use of high-order basis functions permits very large patch sizes. Without an accurate surface, one is constrained to use smaller patches than necessary in order to keep the surface representation error within the desired bounds. Addition of various surface models is straightforward in FastScat because of its object-oriented design and implementation. We will illustrate this with results for scattering from a 1λ radius circle and a 1λ diameter sphere.

Circle

We computed bistatic scattering from a 1λ radius circle with three patch shapes:

- flat line segments,
- quadratic splines,
- and exact arcs.

We made other sources of error, due to quadratures and current discretization, negligible.

The results using flat patches are shown in Figure 5.21. With more than one basis function per patch, the error is entirely dominated by surface modeling. Using twelve patches, the log of the cross section error can not be reduced any lower than -1.7, no matter how high the basis function order. For patches smaller than $\lambda/10$ all benefits of using high-order basis functions are lost. The surface modeling error dominates, even if the lowest order (pulse) basis functions are used.

Figure 5.22 shows the results from a similar calculation using quadratic spline patches instead of flat segments. Much higher accuracy is possible for the same number of unknowns than was the case with the flat patch data shown in Figure 5.21. Nonetheless, the error is still dominated by surface modeling error for more than two basis functions per patch.

Finally, we computed the cross section using the exact surface, each patch being a circular arc. The results in Figure 5.23 show that the computation is no longer limited by surface modeling error. By increasing the number of patches or number of unknowns per patch, one can achieve an accuracy limited only by machine precision.

5.1.6 Fast Multipole Method

We have already reported[HOS+95c] our verification that the use of the FMM can significantly reduce both the time and memory requirements for large $3d$ vector scattering problems. Testing was performed on both spheres and cylindrical rods with hemispherical endcaps. The FMM was compared against both

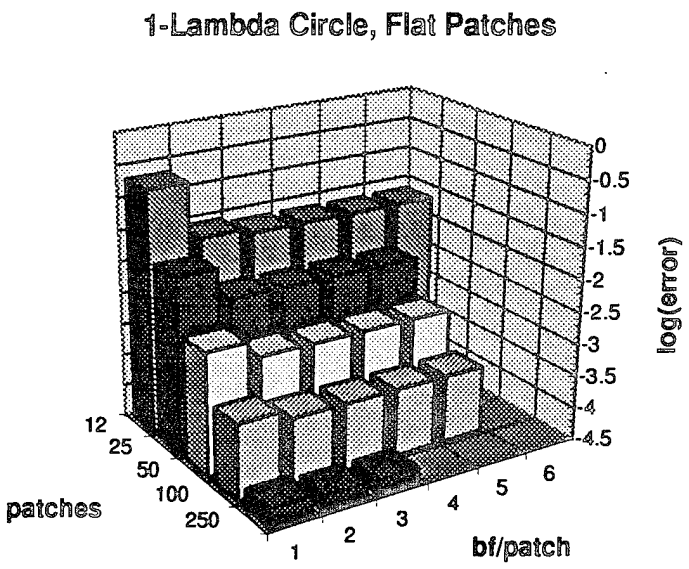

Figure 5.21: Cross section error computing scattering from a 1λ radius circle with Dirichlet boundary conditions. The circle was modeled using line segments for patches.

direct solution and dense iterative solution methods. The dense iterative solver is faster and uses less memory for small objects, but quickly gives way to the FMM for both rods and spheres with major dimension in excess of a few wavelengths. As is expected, the direct method is always slower and uses nearly the same amount of memory as the dense iterative solver.

5.2 Benchmarks

FastScat performance, as measured in CPU time, memory usage and accuracy, has been benchmarked against other RCS prediction codes, in particular RAM2D and CARLOS-3D which are distributed by the Electromagnetic Code Consortium as their benchmark $2d$ and $3d$ method of moments codes respectively. This work was completed before the beginning of this final contract year and is reported in last years annual report[HOS+95c]. Data taken this year on long cylindrical rods, used principally to evaluate the FMM, provided another opportunity to compare FastScat with CARLOS-3D. These results are given at the end of this section.

FastScat was also compared directly with a number of other RCS codes, representing a variety of solution methods, when run on a suite of two-dimensional

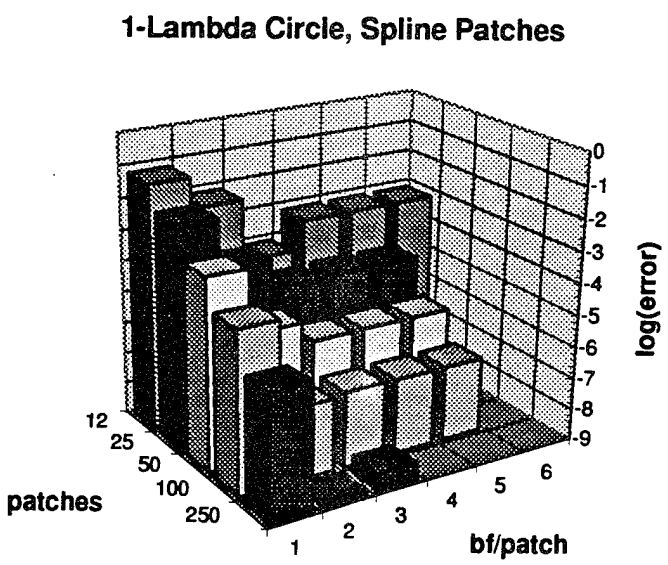

Figure 5.22: Cross section error computing scattering from a 1λ radius circle with Dirichlet boundary conditions. The circle was modeled using quadratic splines for patches.

problems at the GWU workshopB. FastScat's high-order methods allowed us to achieve much higher accuracies using less memory and often with faster solution times.

Rod

A benchmark geometry that we used this year to evaluate FastScat was a rod consisting of a 0.1λ radius cylindrical tube with hemispherical end-caps. Rods of various lengths from 0.5λ to 60λ were used to demonstrate very clearly the advantage of the FMM over dense iterative solution methods and CARLOS-3D in particular, with respect to the amount of memory required to achieve a given solution accuracy. Since all of these codes require in-core setup and solve and single processor machines, no problem could be solved whose requirements exceeded 500Mbyte, the current limit on our workstations. This places a stringent limit on the size of realistic bodies that CARLOS-3D can handle. The short-wavelength regime can barely be reached. In contrast, FastScat using a dense solution method could tackle problems an order of magnitude larger, the FMM improving this by an additional order of magnitude. This is a clear testament to the use of high-order methods in solving short-wavelength scattering and radiation problems and a demonstration of the success of the FMM which comes

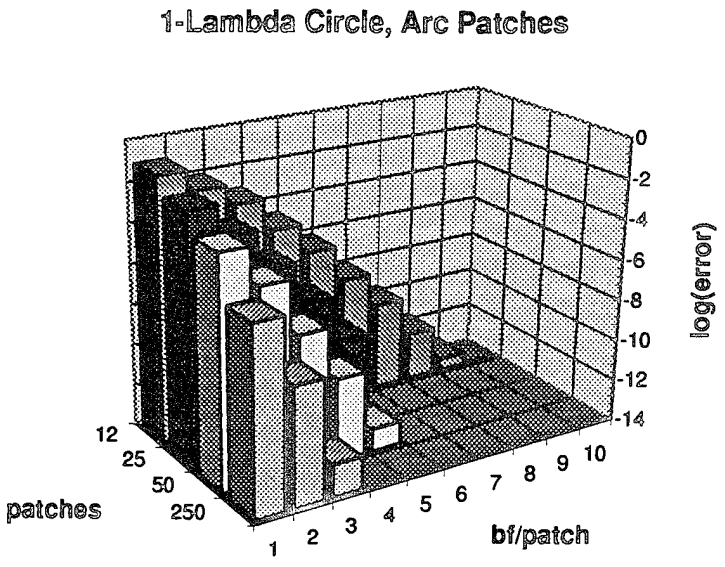

Figure 5.23: Cross section error computing scattering from a 1λ radius circle with Dirichlet boundary conditions. The circle was modeled using circular arcs as patches.

to dominate only in this limit.

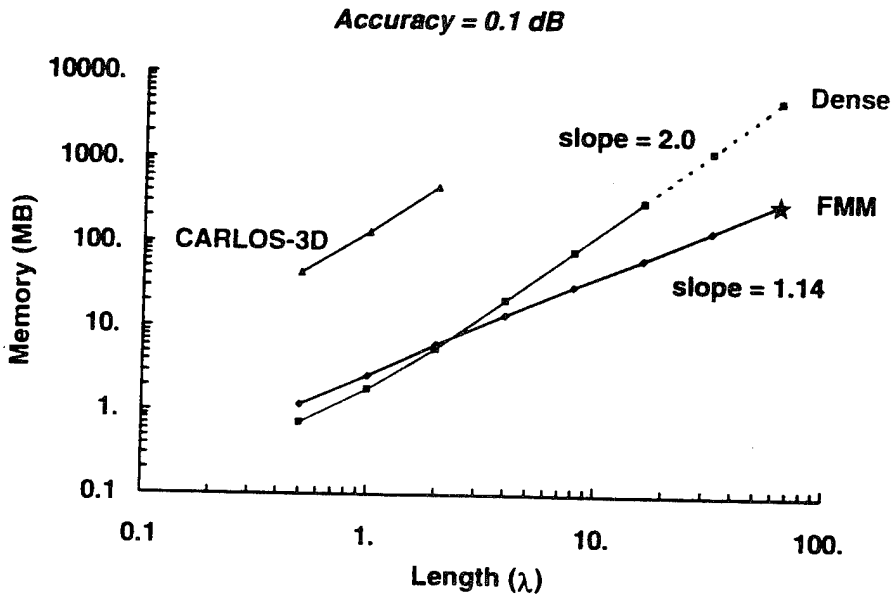

Figure 5.24: Memory (Mbyte) required to achieve $0.1dB$ solution accuracy for various length rods with a fixed 0.1λ radius using CARLOS-3D, FastScat dense solve and FastScat's FMM. The last two points on the dense line are estimates of the memory required, no computation was actually performed.

Appendix A

Reprints

- *The Fast Multipole Method for Electromagnetic Scattering Calculations* [CRW93b]
- *Faster Single-Stage Multipole Method for the Wave Equation* [CRW94]
- *FastScat: An Object-Oriented Program for Fast Scattering Computation* [HST+93a]
- *3D Method of Moments Scattering Computations Using the Fast Multipole Method* [HMS+94b]
- *Electromagnetic Scatering Computations using High-Order Basis Functions in the Method of Moments* [HMS+94a]
- *Surface Modeling in C++* [HST+94]
- *The Fast Multipole Method for Periodic Structures* [RW94]

THE FAST MULTIPOLE METHOD FOR ELECTROMAGNETIC SCATTERING CALCULATIONS †

Ronald Coifman Vladimir Rokhlin
Fast Mathematical Algorithms and Hardware Corp.

Stephen Wandzura*
Hughes Research Labs

January 4, 1993

Abstract

The fast multipole method (FMM) provides a sparse decomposition of the dense impedance matrix obtained by the use of the method of moments in the solution of boundary integral equations. The consequent reduction in computational complexity will allow accurate numerical modeling of far larger electromagnetic scattering and radiation problems than is now possible. We give an elementary derivation and physical interpretation of the FMM for three dimensional electromagnetic problems.

1 Introduction

This talk will give a practical, but not rigorous, exposition of the Fast Multipole Method (FMM). The aim is to give the computational engineer a sufficiently clear understanding of the method to implement it with a minimum of difficulty. For mathematical background and rigor, we refer the reader to Rokhlin's papers[1, 2].

The FMM provides an efficient mechanism for the numerical convolution of the Green function for the Helmholtz equation with a source distribution. It can be used to radically accelerate the iterative solution of boundary integral equations.

We start by comparing the FMM with the fast Fourier transform. For clarity, we then consider the MoM for the scalar wave equation with Dirichlet boundary conditions on the surface of a scatterer. We then show how the presciption is efficiently extended for application to electromagnetic scattering problems. Finally we give physical interpetation and conclusions.

2 Comparison with the Fast Fourier Transform (FFT)

In calculating radar scattering cross sections using the method of moments[3, 4, 5] (MoM) one encounters matrix equations of the form[6]

$$Z \cdot \mathbf{I} = \mathbf{V}, \tag{1}$$

†This research was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contracts No. F49620-91-C-0064 and F49620-91-C-0084. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereron.

where Z is the impedance matrix, I is the unknown current vector (representing currents on the surface of the body), and V is the excitation vector (representing the tangential component of the incident plane wave). The dot product represents an integral over the surface of the scatterer. The FMM provides a sparse decomposition of the matrix Z that can be compared to the FFT. A discrete Fourier transform (DFT) of an N-element vector amounts to the multiplication by a dense $N \times N$ matrix M. Algebraic properties of the DFT allow a decomposition of M into a product of sparse matrices:

$$M = M_1 M_2 ... M_n \,, \tag{2}$$

where n is the number of prime factors of N. In the case of the FMM, *analytic* properties of Z admit a sparse decomposition of the form

$$Z = Z' + V^{\dagger} T V \,, \tag{3}$$

where all matrices on the right hand side are sparse. This decomposition reduces the CPU time and memory required for application of Z to an arbitrary vector from $\mathcal{O}\left(N^2\right)$ to $\mathcal{O}\left(N^{3/2}\right)$. For very large problems, recursive application of the decomposition to Z' yields a $N \log N$ method. However, even for problems that have an order of magnitude more variables than those currently tractable using dense matrix techniques ($N \approx 10^5$), we estimate that the performance of the single-stage algorithm should be near optimal.

3 FMM — Scalar Scattering

For scalar scattering, the matrix elements of Z are given by

$$Z_{nn'} = -i \int_S d^2\mathbf{x} \int_S d^2\mathbf{x}' \, f_n(\mathbf{x}) \frac{e^{ik|\mathbf{x}-\mathbf{x}'|}}{4\pi\,|\mathbf{x}-\mathbf{x}'|} f_{n'}(\mathbf{x}') \,, \tag{4}$$

where $\{f_n\}$ are the basis functions (we naturally assume the use of the Galerkin method) and S is the scatterer surface. By using the elementary identities:

$$\frac{e^{ik|\mathbf{X}+\mathbf{d}|}}{|\mathbf{X}+\mathbf{d}|} = ik \sum_{l=0}^{\infty} (-)^l (2l+1) j_l(kd) h_l^{(1)}(kX) P_l(\hat{d}\cdot\hat{X}) \tag{5}$$

and

$$\int d^2\hat{k}\, e^{i\mathbf{k}\cdot\mathbf{d}}\, P_l(\hat{k}\cdot\hat{X}) = 4\pi i^l j_l(kd) P_l(\hat{d}\cdot\hat{X}) \,, \tag{6}$$

one can show that

$$Z_{nn'} \approx \frac{k}{(4\pi)^2} \int_S d^2\mathbf{x}\, f_n(\mathbf{x}) \int_S d^2\mathbf{x}'\, f_{n'}(\mathbf{x}') \int d^2\hat{k}\, e^{i\mathbf{k}\cdot(\mathbf{x}-\mathbf{x}'-\mathbf{X})} \mathcal{T}_L(kX, \hat{k}\cdot\hat{X}) \,, \tag{7}$$

where

$$\mathcal{T}_L(\kappa, \cos\theta) \equiv \sum_{l=0}^{L} i^l (2l+1) h_l^{(1)}(\kappa) P_l(\cos\theta) \,. \tag{8}$$

For $X \gg |\mathbf{x}-\mathbf{x}'-\mathbf{X}|$, high (e.g. machine) precision can be obtained with a modest value of $L \approx kD$, where D is the maximum value of $|\mathbf{x}-\mathbf{x}'-\mathbf{X}|$. This formula thus provides for the second term of the decomposition Eq. (3) for basis functions of well separated local support. The matrix elements of V and T are given by

$$V_{m\alpha}(\hat{k}) = \int d^2\mathbf{x}\, e^{i\mathbf{k}\cdot\mathbf{x}} f_{n(m,\alpha)}(\mathbf{x}) \,, \tag{9}$$

and

$$T_{mm'}(\hat{k}) = \frac{ke^{-i\mathbf{k}\cdot\mathbf{X}_{mm'}}}{(4\pi)^2}\sum_{l=0}^{L} i^l(2l+1)h_l^{(1)}(kX_{mm'})P_l(\hat{k}\cdot\hat{X}_{mm'})\,. \tag{10}$$

The first term Z' of Eq. (3) represents interactions between "nearby" basis functions for which the FMM cannot be applied. By virtue of the small number of near neighbors, Z' is sparse.

4 FMM — Electromagnetic Scattering

In the solution of the electric field integral equation, the impedance matrix elements take the form[6]

$$Z_{nn'} = -i\sum_{j,j'=1}^{3}\int_S d^2\mathbf{x}\int_S d^2\mathbf{x}'\, f_{nj}(\mathbf{x})G_{jj'}(\mathbf{x}-\mathbf{x}')f_{n'j'}(\mathbf{x}')\,, \tag{11}$$

where

$$G_{jj'}(\mathbf{x}-\mathbf{x}') = \left(\delta_{jj'} - \frac{1}{k^2}\frac{\partial}{\partial x_j}\frac{\partial}{\partial x'_{j'}}\right)\frac{e^{ik|\mathbf{x}-\mathbf{x}'|}}{4\pi\,|\mathbf{x}-\mathbf{x}'|}\,, \tag{12}$$

and the indices j, j' label Cartesian components. By combining the identities above and differentiating with respect to $\mathbf{d}$, one gets

$$\begin{aligned} Z_{nn'} \;\approx\; & \frac{k}{(4\pi)^2}\sum_{j,j'=1}^{3}\int_S d^2\mathbf{x}\, f_{nj}(\mathbf{x})\int_S d^2\mathbf{x}'\, f_{n'j'}(\mathbf{x}') \\ & \int d^2\hat{k}\,\left(\delta_{jj'} - \hat{k}_j\hat{k}_{j'}\right)e^{i\mathbf{k}\cdot(\mathbf{x}-\mathbf{x}'-\mathbf{X})}T_L(kX,\hat{k}\cdot\hat{X})\,, \end{aligned} \tag{13}$$

Now it can be easily seen that the scalar prescription can be modified to an electromagnetic one by promoting $V_{m\alpha}$ to a three dimensional vector, with

$$\mathbf{V}_{m\alpha}(\hat{k}) = \int d^2\mathbf{x}\, e^{i\mathbf{k}\cdot\mathbf{x}}\left[\mathbf{f}_{n(m,\alpha)}(\mathbf{x}) - \hat{k}\hat{k}\cdot\mathbf{f}_{n(m,\alpha)}(\mathbf{x})\right]\,. \tag{14}$$

5 Physical Interpretation

The physics of the FMM rests on the following fact: given a field $\psi(\mathbf{x})$ that satisfies the wave equation

$$(\nabla^2 + k^2)\,\psi(\mathbf{x}) = 0 \tag{15}$$

for all $\mathbf{x}$ outside a given sphere, the field can be reconstructed everywhere outside that sphere from its far field[7, 8].

This means that if the field is radiated by a source density $\rho(\mathbf{x})$, supported only within a sphere of radius R centered at the origin:

$$\phi(\mathbf{x}) = \int_{x'<R} d^3\mathbf{x}'\,\frac{e^{ik|\mathbf{x}-\mathbf{x}'|}}{4\pi\,|\mathbf{x}-\mathbf{x}'|}\rho(\mathbf{x}')\,, \tag{16}$$

then the contribution of the "off-shell" ($q^2 \neq k^2$) components in the Fourier expansion of the Green function[9],

$$\frac{e^{ik|\mathbf{x}-\mathbf{x}'|}}{4\pi\,|\mathbf{x}-\mathbf{x}'|} = \int\frac{d^3\mathbf{q}}{(2\pi)^3}\frac{e^{i\mathbf{q}\cdot(\mathbf{x}-\mathbf{x}')}}{q^2-k^2-i\epsilon}\,, \tag{17}$$

are determined for $x > R$ (after integration over $d^3\mathbf{x}'$) by the radiation condition and the "on-shell" components. The on-shell components, coming from the residue of the pole at $q^2 = k^2$, give the imaginary part of the Green function and the off-shell components give the real part. It is important that the off-shell part is *not* determined by the on-shell part for $x' < R$. This is related to the divergence of the series Eq. (5) for $d > X$. This interpretation clarifies why one only need keep two components in $\mathbf{V}$ for the electromagnetic case; the electromagnetic far field is transverse and has only two independent components.

6 Conclusion

Present methods for computing radar and other scattering cross sections are limited by computer processing and memory requirements. The significance of the increase in problem size made possible by the FMM can be illustrated by considering accurate calculation of RCS for X-band radar. With current methods, the size of the largest body that can be accurately modeled is about a foot. With the same computing power, the techniques that we have described will increase this to 10–100 feet. Such computational capability would significantly reduce the technological risk of expensive projects employing stealth technology. They may likewise revolutionize other applications of scattering computations, such as high-frequency circuit modeling, sonar, and geophysical applications.

References

[1] Vladimir Rokhlin. Rapid solution of integral equations of scattering theory in two dimensions. *Journal of Computational Physics*, 86(2):414–439, 1990.

[2] Vladimir Rokhlin. Diagonal form of translation operators for the Helmholtz equation in three dimensions. Technical Report YALEU/DCS/RR-894, Yale University, Department of Computer Science, March 1992.

[3] Roger F. Harrington. Origin and development of the method of moments for field computation. *IEEE Antennas and Propagation Society Magazine*, pages 31–35, June 1990.

[4] Roger F. Harrington. *Field Computation by Moment Methods.* Macmillan, New York, 1968.

[5] Robert C. Hansen, editor. *Moment Methods in Antennas and Scattering.* Artech, Boston, 1990.

[6] Sadasiva M. Rao, Donald R. Wilton, and Allen W. Glisson. Electromagnetic scattering by surfaces of arbitrary shape. *IEEE Transactions on Antennas and Propagation*, AP-30(3):409–418, May 1982.

[7] A. J. Devaney and Emil Wolf. Radiating and nonradiating classical current distributions and the fields they generate. *Physical Review*, D8(4):1044–1047, August 1973.

[8] Rudolf Peierls. *Surprises in Theoretical Physics.* Princeton Series in Physics. Princeton University Press, Princeton, New Jersey, 1979.

[9] George Arfken. *Mathematical Methods for Physicists.* Academic Press, New York, second edition, 1970.

Faster Single-Stage Multipole Method for the Wave Equation *

Ronald Coifman
Vladimir Rokhlin
Fast Mathematical Algorithms and Hardware Corp.

Stephen Wandzura
Hughes Research Labs

Abstract

The fast multipole method (FMM) provides a sparse decomposition of the impedance matrix arising from a discretization of an integral equation equivalent to the wave equation with radiation boundary condition. Mathematically, the sparse factorization is made possible by a diagonal representation of translation operators for multipole expansions. Physically, this diagonal representation corresponds to the complete determination of fields in the source-free region by the far fields alone.

Because the diagonal form of the translation operator is not a well behaved function, it must be filtered in numerical practice. (This does not constitute a practical limitation to the accuracy of the results obtained with the method because of the superalgebraic convergence of the multipole expansions.) In the originally published version of the FMM, the filtering was accomplished by a simple truncation of the

*This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contracts No. F49620-91-C-0064 and F49620-91-C-0084. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereron.

multipole expansion of the translation operator. This sharp cutoff results in an oscillatory transfer function that is non-negligible over the entire unit sphere (i.e., in all far-field directions). Physically, the transfer function represents the effect a bounded source has on a well-separated observation region, expressed in terms of the far field of the source. This suggests that a suitable transfer function might be non-negligible only in the direction of the separation vector. It turns out that such a transfer function may be obtained by applying a smooth cutoff to the multipole expansion. Although such a transfer function requires the tabulation of far fields in a denser set of directions, the overall computational and storage requirements for a single-stage FMM are reduced to $O(N^{4/3})$ from $O(N^{3/2})$.

1 Review of FMM

The fast multipole method (FMM) for the wave equation[1, 2] gives a prescription for a sparse decomposition of the (impedance) matrix obtained by discretization of the integral kernel

$$G(\mathbf{x}-\mathbf{x}') = \frac{e^{ik|\mathbf{x}-\mathbf{x}'|}}{4\pi\,|\mathbf{x}-\mathbf{x}'|}\,. \tag{1}$$

Mathematically, this decomposition ensues from the diagonal form of the translation operator in the far-field representation[3]. For brevity, this summary relies heavily on the exposition and notation of [2].

Briefly, the FMM works by decomposing the interactions into near-field and far-field parts. This is done by dividing the scatterer into groups and classifing each pair of groups as near or far. The matrix representing the near-field part is sparse by virtue of locality. The far-field part may be factored by using

$$\frac{e^{ik|\mathbf{X}+\mathbf{d}|}}{|\mathbf{X}+\mathbf{d}|} \approx \frac{ik}{4\pi}\int d^2\hat{k}\, e^{ik\cdot\mathbf{d}}\mathcal{T}_L(kX, \hat{k}\cdot\hat{X})\,, \tag{2}$$

where the $\mathcal{T}$ is the diagonal representation of the translation operator:

$$\mathcal{T}_L(\kappa, \cos\theta) \equiv \sum_{l=0}^{L} i^l(2l+1)h_l^{(1)}(\kappa)P_l(\cos\theta)\,, \tag{3}$$

and X is the distance between the two members of a group pair. In the previously published version of the FMM, the sharp cutoff at $l = L$ caused the transfer function $\mathcal{T}$ to be non-negligible over a wide range of angle. As we show below, examination of $\mathcal{T}$ reveals that it may be modified so that it has support only in a narrow range of $\cos\theta$ near 1. The only cost of this modification is a denser sampling of far-field radiation patterns from the groups.

2 The Translation Operator

The transfer function $\mathcal{T}_L(\kappa, \cos\theta)$ represents the interaction between bounded source distributions separated by distance κ/k (where k is the free-space wavenumber) and θ is the angle between the displacement vector of the centers of the groups and a direction at which the far-field of the source distribution is computed. Since we expect the fields radiated from a bounded region to a well separated observation region to be given only in terms of the far-field in directions that point toward the observation region, we might expect that $\mathcal{T}_L(\kappa, \cos\theta)$ would be strongly peaked for $\cos\theta \approx 1$. Furthermore, since convergence of the multipole expansions requires $L \approx kD$, where D is the diameter of the regions, we might also expect that the peak have a width $\delta\theta \propto L/\kappa$. Numerical examination of $\mathcal{T}$ reveals that this is indeed the case; however, there are rather large oscillatory tails outside the peak. In Figure 1, $\mathcal{T}_{10}(30, \cos\theta)$ is plotted. This is the transfer function that one would use for rather small (compared to a wavelength) groups separated by 4.8 wavelengths. The oscillatory tails are reminiscent of *leakage* in power spectrum estimation using the FFT[4]. This suggests that by using a smooth "window function" to compute $\mathcal{T}$ rather than a sharp cutoff, that leakage to large angles may be reduced. In fact, this is the case; even a simple-minded cosine window function, giving

$$\tilde{\mathcal{T}}_L(\kappa, \cos\theta) = \mathcal{T}_L(\kappa, \cos\theta) + \sum_{l=L+1}^{2L} i^l (2l+1) \left[1 - \sin^2 \frac{(l-L)\pi}{2L}\right] h_l^{(1)}(\kappa) P_l(\cos\theta), \quad (4)$$

produces the localized transfer function plotted in Figure 2. Naturally, because we are taking more terms in the multipole expansion of $\mathcal{T}$, we must

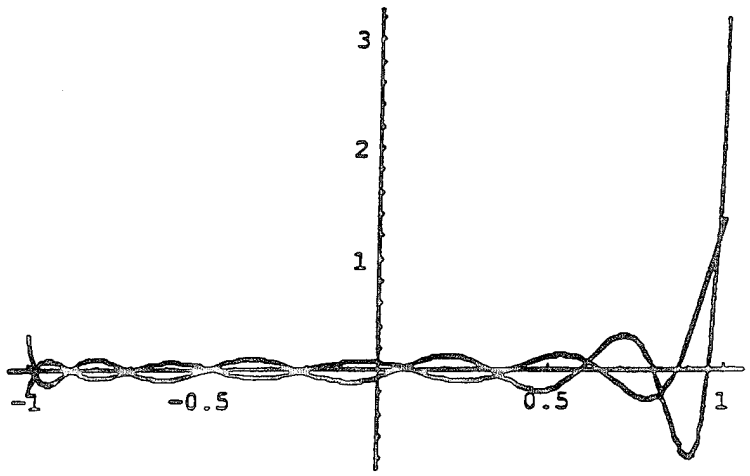

Figure 1: Real and imaginary parts of transfer function $\mathcal{T}$ of $\cos\theta$ for $L = 10$, $\kappa = 30$.

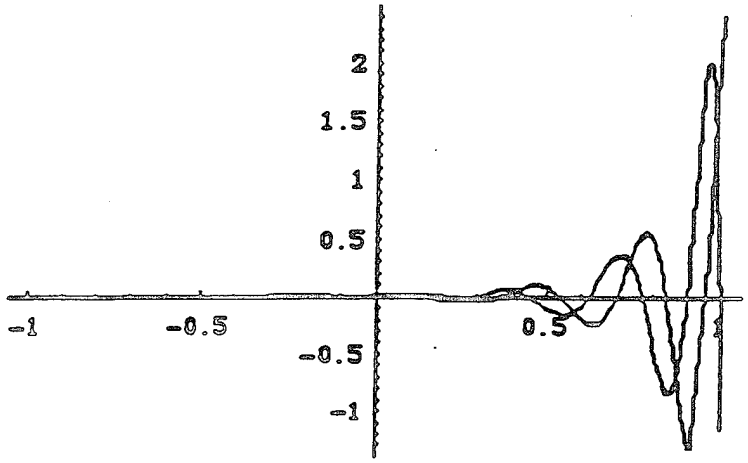

Figure 2: Real and imaginary parts of the localized transfer function $\bar{\mathcal{T}}$ of $\cos\theta$ for $L = 10$, $\kappa = 30$.

sample the far fields in a denser set of directions appropriate to a quadrature rule for spherical integrations exact for a larger set of spherical harmonics. The trigonometric window function in Eq. (4) is only for purposes of illustration; more efficient windows should be used in practice.

3 Complexity Reduction

A detailed analysis, to be published elsewhere, reveals that the window function of l can be chosen to minimize the support in solid angle of $\mathcal{T}$. This analysis confirms the intuition, implied above, that the solid angle of support of the resulting transfer function is about $\pi(kD)^2/(4\kappa^2)$, where D is the diameter of the groups. In the $\mathcal{O}\left(N^{3/2}\right)$ FMM, the operation count of the translation operator application is $\propto KM^2$, where M is the number of groups and K is the number of far-field directions tabulated. It might now seem that this count should be multiplied by a factor $\propto (kD)^2/(4\kappa^2) \propto 1/M$, giving a total count $\propto (K/M)M^2 \propto N$, which is independent of M. This is incorrect, however, because it implies that by decreasing the size of the groups that the number of directions at which the far-field is used can be reduced without limit. Actually, since we must know the far-field of each group in at least one direction for each other group, the number of directions must go to a constant for very small groups. The total operation count for application of the translation operators is thus $(bN/M^2 + c)\, M^2$, where b and c are implementation dependent constants. (Actually, a more careful analysis gives a factor of $\ln M$ in the b term, but it has no effect on the behavior for large N.) Minimizing the sum of this with the operation count for the other steps in the FMM (aN^2/M, where a is another constant), one sees that, for large problems, b is irrelevant, and the total operation count is minimized by choosing

$$M = \left(\frac{aN^2}{2c}\right)^{1/3}, \tag{5}$$

so that the total operation count is $\mathcal{O}\left(N^{4/3}\right)$. For smaller problems, where the c term does not dominate, the operation count varies roughly as $N \ln N$.

References

[1] V. Rokhlin, "Rapid solution of integral equations of scattering theory in two dimensions," *Journal of Computational Physics*, 86(2):414–439, 1990.

[2] R. Coifman, V. Rokhlin, and S. Wandzura, "The fast multipole method: A pedestrian prescription," *IEEE Antennas and Propagation Society Magazine*, 35(3):7–12, June 1993.

[3] V. Rokhlin, "Diagonal form of translation operators for the Helmholtz equation in three dimensions," *Applied and Computational Harmonic Analysis*, 1(1):82–93, December 1993.

[4] W. H. Press, B. P. Flannery, S. Teukolsky, and W. T. Vetterling, *Numerical Recipes — The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1986.

FastScat™: An Object-Oriented Program for Fast Scattering Computation

LISA HAMILTON, MARK STALZER, R. STEVEN TURLEY, JOHN VISHER, AND STEPHEN WANDZURA

Hughes Research Laboratories, 3011 Malibu Canyon Road, Malibu, CA 90265

ABSTRACT

FastScat is a state-of-the-art program for computing electromagnetic scattering and radiation. Its purpose is to support the study of recent algorithmic advancements, such as the fast multipole method, that promise speed-ups of several orders of magnitude over conventional algorithms. The complexity of these algorithms and their associated data structures led us to adopt an object-oriented methodology for FastScat. We discuss the program's design and several lessons learned from its C++ implementation including the appropriate level for object-orientedness in numeric software, maintainability benefits, interfacing to Fortran libraries such as LAPACK, and performance issues.

1 INTRODUCTION

Current problems of interest in computational electromagnetics include the prediction of radar cross sections and the modeling of antenna radiation patterns (see Fig. 1). Methods for computing electromagnetic scattering and radiation generally involve the solution of a matrix equation derived from the discretization of an appropriate integral equation [1]. The matrix equation is often written $Z \cdot I = V$, where the impedance matrix Z depends on the geometry and composition of the scattering or radiating surface, I is a vector containing the expansion coefficients of the current density over the surface, and the excitation vector V represents a dual expansion of the current. The number of unknowns, N, required for accurate modeling of such problems is very large, and, in the past, has severely limited problem size and solution accuracy.

There are two primary areas of difficulty in conventional solutions of these problems. The first is accurate computation of the Z matrix elements. In general, each element of the $N \times N$ matrix requires numeric integration of a function that is often singular on portions of the surface. The second difficulty is the actual solution of such a large matrix equation. This has been done by direct decomposition of the sometimes ill-conditioned Z matrix ($\mathcal{O}(N^3)$ time), or alternatively by iterative methods requiring repeated matrix-vector multiplications ($\mathcal{O}(N^2)$ time for each step).

Recently, a technique called the fast multipole

Received April 1993
Revised June 1993

FastScat™ is a trademark of Hughes Aircraft Company.

This research was partially supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-91-C-0064. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereron.

Scientific Programming. Vol. 2, pp. 171–178 (1993)
CCC 1058-9244/94/040171-08

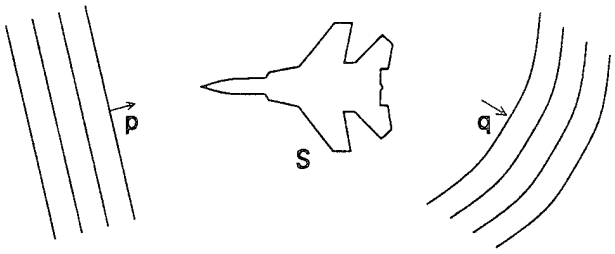

FIGURE 1 Model scattering problem. An incident plane wave $\vec{p}$ (excitation) induces a current on S which re-radiates as the scattered wave $\vec{q}$.

method (FMM) was discovered, which essentially factors the Z matrix into sparse components [2–5]. With this representation, the matrix-vector multiplications required by iterative solvers can be done in $\mathcal{O}(N \log N)$ time. Thus, total solution time is greatly reduced, allowing the study of much larger objects.

Our ongoing effort is to develop a code capable of accurately computing scattering and radiation from surfaces of arbitrary shape and size, represented in either two or three dimensions. In this program, called FastScat, we are implementing conventional solution techniques as well as new computational algorithms, such as the FMM. We also plan to incorporate the ability to scatter from dielectrics and other materials, and to efficiently treat periodic bodies. In addition, FastScat is being used as a testbed to determine the effectiveness of various enhancements such as more accurate surface models, higher order expansion (basis) functions, and more accurate quadrature rules.

To support this work, FastScat must be written in such a way as to be highly modifiable and extensible, as well as reasonably efficient. Specifically, we require a design methodology and language support that can provide a clear implementation of the algorithms and a sensible structure for the underlying data. Our experience in modifying an existing program written in Fortran demonstrated that this, mostly procedural, code lacked important elements needed to incorporate the features described above. Instead, we have turned to an object-oriented methodology [6] in which features such as inheritance, data encapsulation, polymorphism, and dynamic binding allow the key elements of the problem to be expressed and manipulated in a more natural way.

2 AN OBJECT-ORIENTED DESIGN

The design of FastScat is based on the key abstractions of the physics of scattering. In the object-oriented paradigm, a class is used to define a new data type and encapsulates not only the operations that can be performed on that type (methods), but also the implementation or actual data structure of the type. Defining classes to model the physics of the problem provides a clear mapping of the theory and algorithms onto the resulting computer code. For example, the FastScat classes `Surface`, `Z_Matrix`, `Current`, and `Excitation` come directly from the problem formulation given in Section 1. Once defined and implemented, the manipulation of these new types is straightforward and can closely resemble the original equations from physics, thus improving code readability. Using this approach, we have found that when a new class or method seemed awkward or difficult to add it often did not adequately model the physics. As an added benefit of object-oriented thinking, we have sometimes gained a better understanding of physical or theoretical relationships in the problem. On occasion, difficulties in implementation have directed us to a flaw or gap in our physical understanding rather than with the design. The remainder of this section describes some of FastScat's design and the resulting maintainability benefits.

2.1 Modeling Surfaces

In FastScat, the scattering surface or antenna (scatterer) is described using a collection of elementary surfaces. In the current version of FastScat, the elementary surfaces are limited to patches. In two dimensions, a patch is simply a curve in a plane, and in three dimensions, it is a surface. The simplest 3d patch is a flat triangle.

The `Surface` class hierarchy (Fig. 2) provides support for FastScat's surface description. Class `Surface` is abstract and defines basic operations required for all surfaces. These basic operations, which include translate, rotate, scale, and reading/writing, must then be implemented in descendent classes of `Surface`.

A collection of surfaces is maintained by `Composite_Surface`, which descends from `Surface`. An element of a `Composite_Surface` is itself a `Surface`. This organization makes it easy to implement many methods, and permits modeling of hierarchical scatterers. For example, the

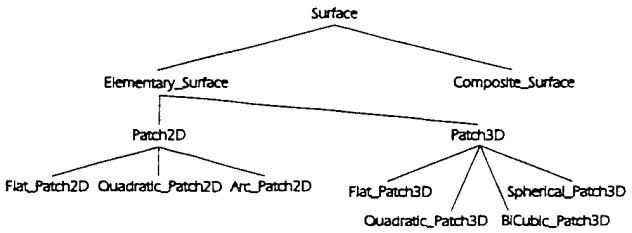

FIGURE 2 Surface class hierarchy.

translate method in `Composite_Surface` simply calls the translate method of each of its elements. An ancillary class supports iteration over all of the elements in a composite surface.

Ultimately, the surface is described in terms of instances of class `Elementary_Surface`. This class, which is derived from `Surface`, currently has two descendants, `Patch2D` and `Patch3D`. In the future, the descendants `Wire2D` and `Wire3D` will be added to support the modeling of wires. The patch classes define several methods. For example, in `Patch2D`, there is a method called `map`, which takes a single parameter $u \in [0, 1]$ and returns a `Vector` to the corresponding point on the patch. The endpoints of the patch are at $u = 0$ and $u = 1$. Another method is `tangent`, which returns the tangent to the patch for a given u. A parallel set of methods is defined by `Patch3D`, except the parameters are u and v. These methods are used by many calculations in FastScat. The important point is that most of the FastScat code is written in terms of `Surface`, `Composite_Surface`, and `Elementary_Surface` objects. The underlying surface model, 2d or 3d, flat, curved, etc., is hidden from most of the code. This eases maintenance and the addition of new features.

2.2 Modeling the Physics

The basic principle behind FastScat's design is to model the physics as closely as possible. The common object-oriented approach is to identify the entities in the problem and proposed solution and to model these using classes. The `Surface` class hierarchy was designed using this approach. Modeling some of the physical concepts is more abstract. Some entities, such as a plane wave, are simple. For a plane wave, we defined a class that contains the wave vector $\vec{k}$ and provides a method to evaluate the wave at any point in space.

A key physics abstraction is a `Surface_Function`. It is defined on the surface of the scatterer and maps a particular location on the scatterer to a tensor. The `Current` and `Excitation` classes are descendants of `Surface_Function`. Various operations are supported on surface functions, including addition, scalar multiplication, and inner product. These operators are used extensively in FastScat's calculations. Although the `Surface_Function` class is currently implemented using class `Array` described in Section 3.2, this representation can and will be changed in the future to implement a different method (Nyström) of discretizing the integral equation.

Closely related to `Surface_Function` is `Surface_Operator`, which maps one surface function onto another. The mapping is performed by the `apply` method. An important example of a `Surface_Operator` is `Z_Matrix` (Z) which takes a `Current` (I) and maps it into an `Excitation` (V). Another example is the FMM, which is implemented in the `FMM` class.

The system $V = Z \cdot I$ can be solved directly using LU decomposition if Z is dense, or by using an iterative solver. Iterative solvers can be used for both dense (`Z_Matrix`) and sparse (`FMM`) surface operators. The iterative solvers are written in terms of `Surface_Operators` and `Surface_Functions`. When support for the FMM was added to FastScat, we only had to concentrate on the details of the FMM as encapsulated by class `FMM`. The solvers did not require modification because they are defined at a higher level of abstraction. The maintainability/extensibility benefits of FastScat's design are discussed further in the next section.

FastScat also contains a class hierarchy for modeling basis functions, which is conceptually similar to that of the surface classes. There is a top-level abstract class `Basis_Function` with descendants for two and three dimensions (`Basis_Function2D` and `Basis_Function3D`). Descendants of these two classes describe particular basis functions, such as Legendre polynomials.

2.3 Maintainability/Extensibility Benefits

One of the major objectives of FastScat was the implementation of the FMM. In the previous section we mentioned how the FMM fit easily into the program's design. This design is also helping to achieve many of FastScat's other objectives. For example, to support different surface models, it is only necessary to add a new descendant to

Patch2D or Patch3D. This flexibility has allowed us to study the importance of higher order surface models for accurate scattering calculations, and has also turned out to be very useful for verification. For a few special geometries, like circles and spheres, the cross section can be computed analytically. In FastScat, a circle can be approximated using flat patches. As the the number of patches increases, so does the solution accuracy. However, even using as many as 1,000 patches only results in a few digits of accuracy in the cross section. Our response was to add a descendant of Patch2D, called Arc_Patch2D, which represents a wedge of a circle. We used the arc patches to construct a perfect model of a circle and were able to compute answers accurate to 11 significant digits.

The structure of the basis function hierarchy allows for similar flexibility. FastScat was originally implemented in terms of pulse (constant) basis functions. Moving up to higher order basis functions was trivial: we simply generalized the pulse basis functions to Legendre polynomials. The rest of the program was unchanged.*

There have been times when it was difficult to use a FastScat component. We have found this with the iterative solvers—they depend on Surface_Function and Surface_Operator, which in turn depend on Surface. Use of the solvers then requires a substantial amount of FastScat code, indicating a flaw in the design. The solvers should have been defined on classes more general than Surface_Function and Surface_Operator, namely Function and Operator. The surface versions would then just be subclasses of the more general versions, and the solvers could be used independently of FastScat by defining the appropriate functions and operators.

3 LESSONS FROM A C++ IMPLEMENTATION

The design of a program is independent of its implementation. In principle, one can have an object-based design and implement it in a traditional language (as is often done with Ada [7]). However, to get full benefit of the methodology, we chose to use an object-oriented language as well.

Pure object-oriented languages, like Smalltalk [8] and CLOS [9], have a high overhead due to their generality, and are not commonly available on supercomputers. C++ [10] has the basic features necessary (such as classes, inheritance, and dynamic binding) for an object-oriented implementation. Because it has been implemented as a translator into C, the language is portable and is widely available on supercomputers. This combination of features and availability led us to the choice of C++.

This section presents some of the lessons we learned from implementing FastScat in C++. Most of what follows is related to performance issues: how to arrange C++ programs so that they run efficiently. We also discuss some of the limitations of C++.

3.1 Overhead of Object-Orientedness

The object-oriented facilities in C++ require runtime support not needed in languages like Fortran. If not properly addressed, this overhead can seriously degrade performance. With our present C++ compiler, the dynamic binding associated with virtual functions takes twice as much time as a regular function call. Consider, for example, the descendants of class Patch2D described previously. Each patch must define the method map, which takes a parameter u and returns a Vector on the surface of the patch. For flat patches, this is a very simple computation and executes in less time than the virtual method call and return. Using inlined methods (type-checked macros) is no help because virtual calls cannot be expanded. However, as illustrated below, there is a simple solution that has the performance of an inline method, the generality of virtual methods, and gives the compiler an opportunity to perform aggressive optimizations.

We often use variations of Gaussian quadrature to perform our integrations. The basic form of a Gaussian quadrature to approximate the integral I of a function $f(x)$ over some region is

$$I \approx \sum_{i=0}^{N-1} w_i f(x_i),$$

where the w_i are weights and the x_i are sampling points (abscissae) for f. Assume we want to integrate the magnitude of the map vector over a

* It is not quite as simple as this. We had to plan ahead and put a method in the basis function class that returns the order of the quadrature required to exactly integrate the function. If we had not, we would have lost accuracy by moving to higher order basis functions.

patch. An obvious C++ implementation is

```
double A1(Patch2D& p) {
  double sum = 0;

  for (int i = 0; i < N; i++)
    sum += w[i]*mag(p.map(x[i]));

  return sum;
}
```

Although simple, this code runs slowly compared with equivalent inlined code due to the overhead in the virtual function call `p.map`. A solution to this problem is to add a `map_all` method that takes a list of places at which to evaluate `map`. The actual implementation is as follows:

```
Class Flat_Patch2D : public Patch2D {
public:
  Vector2D map(double u)
    { return v1 + u*delta; }
  void map_all(int N, double* u,
  Vector2D* results);
  ...
private:
  Vector2D v1, delta;
  ...
};

void Flat_Patch2D::map_all(int N,
double* u, Vector2D* results){
  for (int i = 0; i < N; i++)
    results[i] = map(u[i]);
}
```

The equivalent of function `A1` is then

```
double A2(Patch2D& p) {
  double sum = 0;

  p.map_all(N, x, results);
  for (int i = 0; i < N; i++)
    sum += w[i]*mag(results[i]);

  return sum;
}
```

The loop in `map_all` can execute quickly because `map` can now be expanded. The overhead of the call to `map_all` is negligible because the routine is doing a relatively large amount of work. Higher level code can still be written in terms of the base class `Patch2D` because `map_all` is virtual. Furthermore, the `Vector2D` addition and scalar multiplication can also be expanded. This gives an optimizer or vectorizer all the information it needs (up to aliasing) to generate good code. An additional benefit is that the loop in function `A2` is now far simplier and can be optimized. The performance differences between function `A1` and function `A2` can be dramatic, we saw over a factor of 5 improvement in our quadratures between the two codes, keeping all other conditions constant.

`A2` is slightly more complex than `A1`, primarily due to the fact that some piece of code has to take responsibility for managing `results`. In FastScat we have encapsulated this additional complexity in quadrature classes so that it is completely hidden from the user. Users of our quadrature classes only need to supply "all" versions of methods that are performance critical. For the surface classes, only 4 out of over 20 methods have "all" versions.

By adding some additional methods to our classes, we have kept the benefits of object-oriented programming without sacrificing performance. The moral is to use object-oriented techniques in all but the very small percentage of code that is executed often. Such code must be understandable by the optimizer, meaning that it should be short, and written in terms of fundamental types like int and double. Fortunately, the use of object-oriented techniques allows us to structure the code into easily understood and fast *computational kernels*. The next section discusses this approach further.

3.2 Computational Kernels

FastScat does a significant amount of linear algebra, which is handled by the `Array` and `Matrix` classes. These classes call LAPACK [11] and the BLAS [12, 13] to perform the actual operations. LAPACK, a descendant of LINPACK and EISPACK, is intended to be highly portable and execute efficiently on a large range of target machines. The BLAS is a set of basic linear algebra subprograms, such as matrix-array (vector) multiplication, that are hand tuned to each machine. For example, on a Cray, the BLAS is written to take maximum advantage of the machine's vector units. A good implementation of the BLAS on a scalar machine would ensure that code and data are cached most efficiently and that the execution

of the floating point and integer units is overlapped as much as possible.†

The actual implementation of the Array class is simple:

```
class Array {
public:
  complex dot(Array& b) {
    ZDOTU(&length, data, &stride,
          b.data, &b.stride);
  }
  ...
private:
  complex* data;
  int length, stride;
}
```

The routine ZDOTU is just a BLAS call that does a double precision complex dot product. The stride parameters tell how many elements to skip between consecutive array indices. Note that because dot can be inlined, the users of Array are effectively using the BLAS directly. This illustrates a useful technique: place C++ "wrappers" around high quality libraries implemented in other languages. The libraries then become C++ objects that can be used like any other object.

By carefully isolating the critical code in an application, the performance of an object-oriented program can be made as good as the best programs written in traditional languages. One additional benefit is that the object-oriented code is very portable. Only the kernels might need modification for a particular architecture.

3.3 C++ Limitations

Despite its rich set of features, C++ does have limitations. One that we found particularly frustrating is the lack of multimethods [14, 15], a generalization of virtual methods. A virtual method dynamically dispatches to code, which is selected based on the type of its first argument (this). A multimethod can dispatch on the types of many arguments. Consider a Tensor class that has descendants for Scalars, Vectors, Second-Rank Tensors, and so on, for which we want to define a set of arithmetic operations. The base class Tensor has a virtual method mul(Tensor) that must be defined by each derived class. The problem is in the implementation of mul in the derived classes:

```
Rank2::mul(Tensor& t) {
  select (t.is_a()) {
  case scalar :     // do
Rank2*Scalar
  case vector :     // do
Rank2*Vector
  case rank2  :     // do Rank2*Rank2
  // make higher rank class do the work
  default     : return t.mul(*this); }
}
```

This code is ugly, it cannot be inlined, and using is_a methods to return a type tag is a poor practice. The code is also difficult to maintain, because a class of a given rank must be a friend to all classes having a lower rank (scalar is rank 0, vector is rank 1, etc.). With multimethods, the solution is much cleaner and potentially more efficient because each method is responsible for only one kind of multiplication, for example, Scalar*Vector. Other solutions are possible in C++, but they are all similar in nature and suffer from the problems mentioned. This type of construction arises often in mathematics and it is unfortunate it does not have a clear expression in C++.

A second limitation of C++ is its lack of automatic memory management. Of course, any sort of memory management scheme can be implemented in C++, but we have found that a significant amount of effort goes into designing storage management solutions for various classes, and finding memory leaks. It is common for a C++ program to have several different storage management schemes. For example, in FastScat we use a reference counting technique [15] for the Array class to eliminate unnecessary copying of large objects, and an ownership-based scheme in the Composite_Surface class for patches. Several of the methods in a class (constructors, the destructor, and the assignment operator) must be concerned with memory management. The problem with multiple schemes and methods is that memory management must always be on the mind of the programmer and is a distraction from solving the problem of interest. We believe that some sort of default memory management, which can

† The array and matrix class were originally implemented entirely in C++. The implementation used the standard C convention that the rightmost index varies the fastest. When we switched over to the BLAS, we converted the internal storage format. Although this was a major data representation change, not a single line of code outside the matrix class had to be changed.

be overridden when necessary, would be beneficial.

Finally, C++ tools are still immature. Some vendors have been slow to implement language features, such as templates. Also, the lack of exception handling in most implementations makes error handling clumsy. These problems should disappear with time.

4 CONCLUDING REMARKS

Object-oriented programming is not without costs. We have noticed that it takes more time to design an object-oriented program than a procedural one, which is consistent with some estimates that up to 40% of the effort required to write an object-oriented program goes into the design phase. Also, when object-oriented languages are used in an overly procedural fashion (which is quite easy to do in C++), the benefits of the methodology are lost and the resulting code is often worse than a traditional program. This is similar to an effect noticed when Ada was first introduced. Many programmers were quickly retrained in the Ada syntax but not its design philosophy. Of course, the payback to putting more effort in the design, is in reduced debugging time and easier maintainability/extensibility.

The use of object-oriented languages for numerical applications is being hampered by the fact that object-oriented languages are not Fortran. Fortran is still the language of choice for a majority of people doing computational science, particularly on supercomputers. There are a number of reasons for this:

1. Supercomputer Fortran compilers typically vectorize code better than other compilers.
2. Fortran is widely understood.
3. A great deal of Fortran code exists.
4. There is a built-in resistance to change.

In order for object-oriented design and programming to make serious inroads in computational science, scientists and programmers are going to have to see some obvious benefits. We think the most convincing argument will come from the extensibility of object-oriented programs. If a computational scientist sees a group getting good results quickly, by virtue of being able to easily change their programs, the scientist will naturally become interested in the programming techniques.

In summary, we based the top-level design of FastScat on the physics of scattering. This lead to a flexible code that is easy to maintain and extend, and yet does not necessarily sacrifice efficiency. The fundamental calculations are performed by computational kernels such as the BLAS and a small set of hand-tuned methods in the quadrature classes. The high-level classes simply orchestrate the operation of the kernels. In the future, we plan to extend FastScat to handle more complex scattering problems and to port the code to massively parallel machines and to vector machines such as the Cray.

REFERENCES

[1] R. F. Harrington, *Field Computation by Moment Methods*. New York: Macmillan, 1968.

[2] R. Coifman, V. Rokhlin, and S. Wandzura, "The fast multipole method: A pedestrian prescription," *IEEE Antennas Propagation Soc Magazine*, vol. 35, pp. 7–12, 1993.

[3] V. Rokhlin, "Solution of acoustic scattering problems by means of second kind integral equations," *Wave Motion*, vol. 5, pp. 257–272, 1983.

[4] V. Rokhlin, "Rapid solution of integral equations of scattering theory in two dimensions," *J. Comput. Phys.* vol. 86, pp. 414–439, 1990.

[5] V. Rokhlin, *Diagonal Form of Translation Operators for the Helmholtz Equation in Three Dimensions*. Technical Report YALEU/DCS/RR-894, Yale University, Department of Computer Science, March 1992.

[6] B. Meyer, *Object-Oriented Software Construction*. New York: Prentice Hall, 1988.

[7] G. Booch, *Software Engineering with Ada*. Menlo Park, CA: Benjamin/Cummings, 1983.

[8] A. Goldberg and D. Robson, *Smalltalk-80: The Language and Its Implementation*. Reading, MA: Addison-Wesley, 1983.

[9] S. Keene, *Object-Oriented Programming in Common Lisp*. Reading, MA: Addison-Wesley, 1988.

[10] M. A. Ellis and B. Stroustrup, *The Annotated C++ Reference Manual*. Reading, MA: Addison-Wesley, 1990.

[11] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostouchov, and D. Sorensen, *LAPACK User's Guide*. Philadelphia: Society for Industrial and Applied Mathematics, 1992.

[12] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, "Algorithm 679: A Set of Level 3 Basic

Linear Algebra Subprograms." *ACM Transact. Math. Software*, vol. 16, pp. 18–28, 1990.

[13] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, "Algorithm 656: An extended set of Fortran basic linear algebra subprograms." *ACM Transact. Math. Software*, vol. 14, pp. 18–32, 1988.

[14] R. Agrawal, L. G. DeMichiel, and B. G. Lindsay, *OOPSLA Conference Proceedings*. Reading, MA: Addison-Wesley, 1991.

[15] J. O. Coplien, *Advanced C++: Programming Systems and Idioms*. Reading, MA: Addison-Wesley, 1992.

3D Method of Moments Scattering Computations Using the Fast Multipole Method†

L.R. Hamilton P.A. Macdonald M.A. Stalzer
R.S. Turley* J.L. Visher S.M. Wandzura

Hughes Research Laboratories, Malibu, CA 90265

Abstract

The Fast Multipole Method (FMM) dramatically reduces the time and memory required to compute radar cross sections and antenna radiation patterns compared to dense matrix techniques[1]. We have implemented the FMM in a method of moments (MoM) program to compute electromagnetic scattering from large bodies of arbitrary shape. We compare the memory and time required using the FMM to that for direct and iterative solutions using a dense impedance matrix.

1 Introduction

We will demonstrate the improvement in computation time and memory requirements possible by using the FMM in method of moments calculations of electromagnetic scattering and radiation from objects with surface areas greater than a few wavelengths in size. The scaling of the algorithm is such that it becomes increasingly attractive (compared to dense matrix methods) as the size of the body increases.

We will first present a brief review of the FMM as it applies to 3d electromagnetic scattering. We will then discuss some details of the program in which we implemented the FMM. Finally, we will compare the memory and CPU requirements for computing radar cross sections using the FMM with dense matrix techniques. The three shapes we will use for comparison are spheres, almonds, and cylinders.

2 The Fast Multipole Method

In method of moments solutions to boundary integral equations, one is faced with solving large systems of equations of the form

$$ZI = V\,, \tag{1}$$

where Z is a dense $N \times N$ matrix and I and V are column vectors of length N. N is the number of current expansion (basis) functions. Eq. (1) can be solved by a number of iterative schemes. These techniques involve the computation of the product of Z and a vector $\tilde{I}$ one or more times for each iteration. This operation takes $\mathcal{O}\left(N^2\right)$ operations and usually dominates all other operations in the iterative loop.

In the FMM, one groups the N basis functions into M groups so that the basis functions in each group have neighboring support. For the simplest single stage FMM[1], the optimum value for M is proportional to $\sqrt{N}$. Let the index m run over the groups and the index α

†This research was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract Numbers F49620-91-C-0064 and F49620-91-C-0084. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereron.

refer to a basis function within a particular group. The dense matrix Z is then replaced with the expression

$$Z \approx Z' + \mathbf{V}T\mathbf{V}^{\dagger} , \tag{2}$$

where Z', $\mathbf{V}$ and T are all sparse. Z' are those components of the original Z matrix for interactions between nearby regions of the target (typically within about one wavelength). The approximation can be made arbitrarily precise by the appropriate choice of FMM parameters in the computation of $\mathbf{V}$ and T.

The components of $\mathbf{V}$ are given by

$$\mathbf{V}_{m\alpha}(\hat{k}) = \int d^2\mathbf{x}\, e^{i\mathbf{k}\cdot\mathbf{x}} \left[\mathbf{f}_{m\alpha}(\mathbf{x}) - \hat{k}\hat{k}\cdot\mathbf{f}_{m\alpha}(\mathbf{x})\right] , \tag{3}$$

where $\mathbf{f}$ are the basis functions. $\mathbf{V}$ is evaluated at $K \approx \sqrt{N}$ angles $\hat{k}$ needed for a quadrature over the surface of a sphere.

The sparse matrix T is

$$T_{mm'}(\hat{k}) = \frac{k}{(4\pi)^2} \sum_{l=0}^{L} i^l (2l+1) h_l^{(1)}(kX_{mm'}) P_l(\hat{k}\cdot\hat{X}_{mm'}) , \tag{4}$$

where $X_{mm'}$ is the distance between the centers of the groups m and m'. The number of terms in the sum, L, is chosen to give the desired accuracy in the FMM expansions.

The storage required for Z', $\mathbf{V}$, and T is proportional to $N^{3/2}$. The application of Z to a vector $\tilde{I}$ can be done in $\mathcal{O}\left(N^{3/2}\right)$ operations as follows:

1. Compute the $\mathcal{O}(N)$ vectors

$$\mathbf{s}_m(\hat{k}) = \sum_{\alpha} \mathbf{V}^{\circ}_{m\alpha}(\hat{k}) \tilde{I}_{m\alpha} . \tag{5}$$

   This takes $\mathcal{O}\left(N^{3/2}\right)$ operations.

2. Compute the $\mathcal{O}(N)$ quantities

$$\mathbf{g}_m(\hat{k}) = \sum_{m'} T_{mm'}(\hat{k}) \mathbf{s}_{m'}(\hat{k}) . \tag{6}$$

   This also takes $\mathcal{O}\left(N^{3/2}\right)$ operations.

3. The product $Z\tilde{I}$ can now be computed as

$$(Z\tilde{I})_{m\alpha} \approx \sum_{m'\alpha'} Z'_{m\alpha m'\alpha'} I_{m'\alpha'} + \int d^2\hat{k}\, \mathbf{V}_{m\alpha}(\hat{k}) \cdot \mathbf{g}_m(\hat{k}) . \tag{7}$$

   The sum involves $\mathcal{O}\left(\sqrt{N}\right)$ operations because of the sparsity of the matrix Z'. The numerical evaluation of the integral involves a multiplication and an addition for each of the $K = \mathcal{O}\left(\sqrt{N}\right)$ directions $\hat{k}$. Thus, this final step also takes $\mathcal{O}\left(N^{3/2}\right)$ operations.

3 Implementation

We have included the fast multipole operator as one of two possible representations for impedance operators in our FastScat^{TM} scattering program. The parts of the program involving geometry generation, surface discretization, common matrix element computations, excitation vector computations, solvers, and far field computations are thus identical between the solution techniques we are comparing.

Surface We used exact analytical representations of the surface to avoid errors associated with inadequate geometric representations[2]. Each surface is divided into curved triangular patches.

Basis Functions We used high order basis functions to permit efficient and accurate parameterization of the surface current on arbitrarily large patches. The appropriate number of basis functions per patch is determined by the precision desired in the result and somewhat by the geometry. We utilize the Galerkin technique of using the same basis and testing functions, thus making errors in the computed scattering cross section second order in discretization errors in the surface currents[3].

Quadratures The required numerical integrations are carried out using high-order Gaussian-type quadrature rules. Special rules were developed to handle the various singularities appearing in the integrands. We selected the order of the quadrature rules to give the same precision as that determined by the discretization and FMM expansions.

Massively Parallel Architectures Recently we have started to study the implementation of the FMM on massively parallel architectures such as the Intel Paragon. These machines typically consist of several hundred fast RISC microprocessors interconnected by a communications network. Let $T(1)$ be the time required to compute ZI with one processor and $T(P)$ be the time required with P processors. Ideally, we would like $T(P)/T(1) = O(P)$. This is often difficult or impossible to achieve due to inherently sequential portions of the algorithm and communications costs.

For the FMM, the essential problem is find an optimal distribution of the data structures $\mathbf{s}, \mathbf{g}, \mathbf{V}, I, B, T$, and S. We are exploring two possible distributions:

1. assign one group to each processor, or
2. assign one k direction to each processor.

In both cases, $P = O(\sqrt{N})$, and each node requires $O(N)$ memory and does $O(N)$ floating point operations per calculation of ZI. For distribution 1, the N vectors $\mathbf{s}_m(\hat{k})$ must be communicated between the nodes between steps 1 and 2. For distribution 2, the $N^{3/2}$ *complex numbers* $\mathbf{V}_{m\alpha}(\hat{k}) \cdot g_m(\hat{k})$ must be communicated partway through step 3. However, this communication can be done in $O(N)$ time by using a new communications primitive that we call multi-gather. Furthermore, the communications can be overlapped with the computation of currents due to the nearby interactions in step 3. We expect $T(P)/T(1)$ we be of $O(P)$ for both distributions.

As an example, a 100,000 unknown problem requires a little over 300 processors each with approximately 20Mbytes of memory which is a fairly standard configuration for massively parallel machines. We anticipate that the time per iteration will be on the order of one second.

4 Results

We have compared the memory and time requirements of the FMM and dense matrix approaches for three problems: a sphere, an almond, and a cylinder with hemispherical end caps.

Since the purpose of this paper is to compare the FMM to dense matrix techniques, the most relevant measure of solution time is the time required per iteration using the same iterative solver. The results are then independent of the particular iterative solver one chooses to use. In order to facilitate comparison of FMM results to direct solution techniques, we also compare the total time required to solve the problem with a particular solver using the biconjugate gradient method.

The convergence of the solutions were checked by comparing our results to analytical solutions in the case of the sphere, or to higher accuracy MoM solutions in the cases of the almond, and cylinder. Arbitrarily accurate answers can be obtained by appropriate choices of FMM constants, quadrature orders, and basis function orders.

4.1 Sphere

The first comparison case we computed was for scattering from a perfectly conducting spheres of various radii. The computations were compared to the Mie series to measure the accuracy of the results. We will show the memory and cpu time required for various sized spheres. For the iterative solutions, we will show the error at each iteration and the time required per iteration.

4.2 Almond

The NASA almond consists of a half ogive-ellipse smoothly joined to a half ellipsoid. It is a good test of MoM codes because of the large dynamic range in the monostatic scattering cross section. We computed scattering from the almond at various frequencies for waves incident normal to each tip. We will compare our results to high accuracy MoM solutions and to measured data, where available.

4.3 Cylinder

The final target we used for these comparisons is a cylinder with hemi-spherical end caps. We used a cylinder with a radius of 0.1λ and varying lengths. The results are compared to high accuracy MoM solutions.

5 Conclusion

We have demonstrated the fast multipole method improves memory and cpu requirements for solving large RCS and antenna problems via the method of moments. It becomes increasingly attractive as the size of the scattering problem increases.

References

[1] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. The fast multipole method: A pedestrian prescription. *IEEE Antennas and Propagation Society Magazine*, 35(3):7–12, June 1993.

[2] Lisa Hamilton, Vladimir Rokhlin, Mark Stalzer, R. Steven Turley, John Visher, and Stephen Wandzura. The importance of accurate surface models in RCS computations. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 3, pages 1136–1139, Ann Arbor, MI, June 1993. IEEE.

[3] Stephen M. Wandzura. Optimality of Galerkin method for scattering computations. *Microwave and Optical Technology Letters*, 4(5):199–200, April 1991.

Electromagnetic Scattering Computations using High-Order Basis Functions in the Method of Moments†

L.R. Hamilton P.A Macdonald M.A. Stalzer
R.S. Turley* J.L. Visher S.M. Wandzura

Hughes Research Laboratories, Malibu, CA 90265

Abstract

Using high order basis functions in Galerkin method of moments calculations permits a significant reduction in the number of unknowns needed to achieve a given accuracy in modeling a scatterer or antenna[1]. We have developed a set of vector valued basis functions of arbitrary order with a sparse overlap matrix. We demonstrate the reduction of computational effort for a given accuracy by computing the scattering from a sphere.

1 Introduction

We have demonstrated that using high order basis functions reduces the effort to compute the scattering crossection to a given accuracy[1]. We have developed a set of basis functions that are useful for electromagnetic scattering calculations.

2 Model Problem

The model problem for this study was a PEC sphere. The Electric Field Integral Equation (EFIE) is [3][4]

$$-\hat{n}(\mathbf{x}) \times \mathbf{E}_{\text{inc}}(\mathbf{x}) = \frac{i\omega\mu}{4\pi}\hat{n}(\mathbf{x}) \times \int_S d^2\mathbf{x}'\, \mathbf{J}(\mathbf{x}')G(\mathbf{x},\mathbf{x}') + \frac{1}{k^2}\nabla' \cdot \mathbf{J}(\mathbf{x}')\nabla' G(\mathbf{x},\mathbf{x}')\,, \qquad (1)$$

where

$$G(\mathbf{x},\mathbf{x}') = \frac{e^{ik|\mathbf{x}-\mathbf{x}'|}}{|\mathbf{x}-\mathbf{x}'|}\,. \qquad (2)$$

The basis functions we have developed are generalizations to those of Rao, Wilton and Glisson[3](RWG) and are supported on adjacent triangular patches. We have extended these basis functions to arbitrary order and to allow mapping onto curved surfaces[5]. Using the Galerkin technique, Eq. (1) can be reduced to a matrix equation by approximating J with a linear combination of basis functions $\mathbf{f}_i$, multiplying both sides of Eq. (1) by the same set of basis functions and integrating over $\mathbf{x}$. Letting

$$\mathbf{J}(\mathbf{x}) \approx \hat{\mathbf{J}}(\mathbf{x}) = \sum_i I_i \mathbf{f}_i(\mathbf{x}) \qquad (3)$$

†This research was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract Numbers F49620-91-C-0064 and F49620-91-C-0084. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereron.

$$V_j \equiv \int_S d^2\mathbf{x}\, \mathbf{E}^{\rm inc}(\mathbf{x}) \cdot \mathbf{f}_j(\mathbf{x}) \tag{4}$$

$$Z_{jj'} \equiv \frac{i\omega\mu}{4\pi}\left\{\int_S d^2\mathbf{x} \int_S d^2\mathbf{x}'\, G(\mathbf{x},\mathbf{x}')\mathbf{f}_j(\mathbf{x}) \cdot \mathbf{f}_{j'}(\mathbf{x}') - \frac{1}{k^2}\int_S d^2\mathbf{x} \int_S d^2\mathbf{x}'\, G(\mathbf{x},\mathbf{x}')\left[\nabla \cdot \mathbf{f}_j(\mathbf{x})\right]\left[\nabla' \cdot \mathbf{f}_{j'}(\mathbf{x}')\right]\right\}, \tag{5}$$

where S is the surface of the scatterer, the matrix equation to be solved is

$$ZI = V\,. \tag{6}$$

3 Basis Functions

Our set of basis functions are composed of edge-based and patch-based functions. The edge-based functions, $\mathbf{f}_n^e$, are supported on two patches which share a common edge. The basis function $\mathbf{f}_0^e$ is shown in Fig.2. There are two kinds of patch-based functions, both of which are supported on a single patch. One kind, $\mathbf{f}_n^{pe}$, vanishes on two edges and is tangential to the remaining edge. The other patch function, $\mathbf{f}_{nm}^{p}$, vanishes on all three edges. The basis functions $\mathbf{f}_0^{pe}$, $\mathbf{f}_{00}^{p}$ are shown in Fig. 3. A general triangular patch can be mapped to the equilateral triangle shown in Fig.1. The relationship between x, y coordinates and the patch coordinates u_1, u_2, u_3 is

$$x = \frac{1}{2}(2u_3 - u_1 - u_2) \qquad y = \frac{\sqrt{3}}{2}(u_1 - u_2)\,, \tag{7}$$

where $u_1 + u_2 + u_3 = 1$. The tangent vectors for the patch are given by

$$\mathbf{t}_1 = \left(\frac{\partial \mathbf{x}}{\partial u_1}\right)_{u_3} \tag{8}$$

$$\mathbf{t}_2 = \left(\frac{\partial \mathbf{x}}{\partial u_2}\right)_{u_3}, \tag{9}$$

where $\mathbf{x}$ is the position vector for an arbitrarily shaped patch. For a flat patch, $\mathbf{x} = u_1\mathbf{V}_1 + u_2\mathbf{V}_2 + u_3\mathbf{V}_3$ and $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$ are the vertices of the triangular patch. $P_n(x)$ and $P_n^{\alpha,\beta}(x)$ denote the Legendre and Jacobi polynomials. The nth order edge-based basis function is

$$\mathbf{f}_n^e(\mathbf{x}) = \frac{l_3(1-x)^{n+1}}{\sqrt{g(\mathbf{x})}}\left[\mathbf{t}_1\left\{P_{n+1}(\eta) + P_n(\eta)\right\} - \mathbf{t}_2\left\{P_{n+1}(\eta) - P_n(\eta)\right\}\right] \tag{10}$$

where

$$\eta = \frac{\sqrt{3}y}{1-x}\,, \tag{11}$$

and l_3 is the length of the common edge and $g(\mathbf{x})$ is the determinant of the mapping $u_{1,2} \rightarrow \mathbf{x}$[5]. For each edge, the nth order patch-based basis function $\mathbf{f}_n^{pe}$ is given by

$$\mathbf{f}_n^{pe}(\mathbf{x}) = \frac{l_3(1-x)^{n+1}}{\sqrt{g(\mathbf{x})}}\left[\mathbf{t}_1\left\{P_{n+1}(\eta) + P_n(\eta)\right\} - \mathbf{t}_2\left\{P_{n+1}(\eta) - P_n(\eta)\right\} - 2\,\mathbf{f}_{n+1}^e(\mathbf{x})\right], \tag{12}$$

where l_3 is the length of the edge. The n nth order patch-based functions that vanish on all three edges are given by

$$\mathbf{f}_{nm}^{p\pm}(\mathbf{x}) = \frac{(1-x)^{m+2}}{\sqrt{g(\mathbf{x})}}(\mathbf{t}_1 \pm \mathbf{t}_2)\,\frac{(2n+7)(2x+1)}{3(n-m+1)}P_{n-m}^{(1,2m+5)}\left[\frac{1-4x}{3}\right]\left[P_{m+2}(\eta) - P_m(\eta)\right]. \tag{13}$$

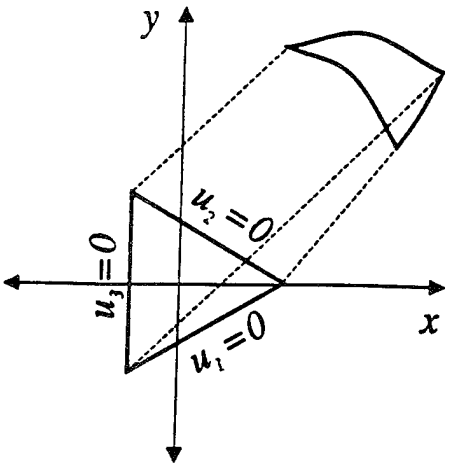

Figure 1: Mapping to an equilateral triangle

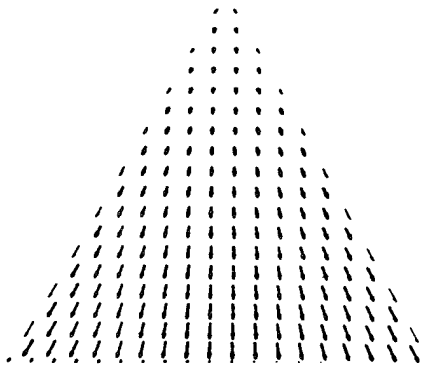

Figure 2: Plot of $\mathbf{f}_0^e$

Surface Divergence These basis functions are identical to the form used in [5]. We can then write the surface divergence as

$$\nabla_S \cdot \mathbf{f}(\mathbf{x}) = \frac{1}{\sqrt{g(\mathbf{x})}} \left[\frac{\partial P^1}{\partial u_1} + \frac{\partial P^2}{\partial u_2} \right] \tag{14}$$

4 Results

We have compared the memory and time requirements for the scattering from a sphere using the RWG basis functions and our high order basis functions. The accuracy of both computations was checked using the Mie series solution. We will show the memory and cpu time required for different basis function orders.

5 Conclusion

We have extended our previous work with high order scalar basis functions to vector basis functions for use in the Galerkin method of moments. The use of these basis functions significantly reduces the time and memory required to obtain an accurate solution to the EFIE.

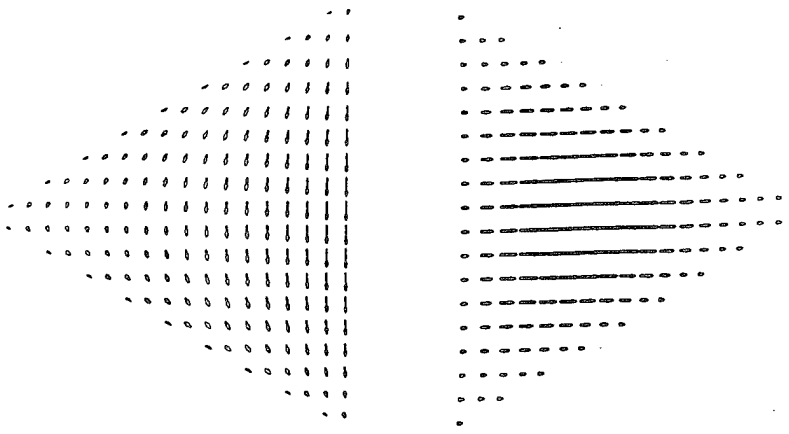

Figure 3: Plot of $\mathbf{f}_0^{p+}$ and $\mathbf{f}_{00}^{pe}$

References

[1] Lisa Hamilton, Mark Stalzer, R. Steven Turley, John Visher, and Stephen Wandzura. Method of moments scattering computations using high-order basis functions. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 3, pages 1132–1135, Ann Arbor, MI, June 1993. IEEE.

[2] Lisa Hamilton, Vladimir Rokhlin, Mark Stalzer, R. Steven Turley, John Visher, and Stephen Wandzura. The importance of accurate surface models in RCS computations. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 3, pages 1136–1139, Ann Arbor, MI, June 1993. IEEE.

[3] Sadasiva M. Rao, Donald R. Wilton, and Allen W. Glisson. Electromagnetic scattering by surfaces of arbitrary shape. *IEEE Transactions on Antennas and Propagation*, AP-30(3):409–418, May 1982.

[4] Andrew J. Poggio and Edmund K. Miller. Integral equation solutions of three-dimensional scattering problems. In Raj Mittra, editor, *Computer Techniques for Electromagnetics*, chapter 4. Hemisphere, New York, 1987. Second printing.

[5] Stephen M. Wandzura. Electric current basis functions for curved surfaces. *Electromagnetics*, 12:77–91, 1992.

Surface Modeling in C++ *

Lisa Hamilton † Mark A. Stalzer R. Steven Turley
John L. Visher

Hughes Research Laboratories
3011 Malibu Canyon Road
Malibu, CA 90265

Abstract

In computational physics, it is often necessary to model surfaces of arbitrary shape and complexity. Collections of connected parametric patches can be used to represent such surfaces. This paper describes the class hierarchy we have developed to model $2d$ and $3d$ surfaces in FastScat‡, a program for computing radar cross sections and antenna radiation patterns. We discuss how an object-oriented design and the features of C++ allowed us to implement a complicated array of surface parameterizations, producing effective code that is easy to use and maintain.

1 Introduction

FastScat is a program for accurately computing scattering and radiation from surfaces of arbitrary shape and size, represented in either two or three dimensions. Its purpose is to support the study of recent algorithmic advancements, such as the fast multipole method [CRW93, HST+93a], which promise speed-ups of several orders of magnitude over conventional algorithms. In addition, FastScat is being used as a testbed to determine the effectiveness of various enhancements such as exact surface models [HRS+93], high order basis functions [HST+93b], and high order quadrature rules.

[HST+93c] is a broad introduction to FastScat and its object-oriented design. Here, we focus on a particular portion of the program, elaborating on the classes used to model surfaces in FastScat. We begin with a brief description of our surface parameterizations and surface code requirements. In Section 3 we describe the design and implementation of the resulting class hierarchy. Finally, in the remaining sections of this paper, we discuss issues of use and efficiency related to the surface code.

*This research was partially supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-91-C-0064. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereron.

†Presenter. Phone: 310-317-5894. Email: hamilton@macaw.hrl.hac.com, FAX: 310-317-5483

‡FastScat is a trademark of the Hughes Aircraft Company.

2 Surface Parameterizations

To model a wide range of complex surfaces (from antennas to missiles and airplanes), FastScat requires a general surface representation. We model surfaces as collections of connected parametric "patches," each closely modeling a portion of the surface. In $2d$, these patches are straight or curved segments defined by a function $\mathbf{x}(u)$, $u \in [0,1]$. For a straight segment, a point on the the patch is given by

$$\mathbf{x}(u) = \mathbf{v}_1 + u(\mathbf{v}_2 - \mathbf{v}_1)\,, \tag{1}$$

where $\mathbf{v}_1$ and $\mathbf{v}_2$ are the endpoints of the segment. A quadratic patch requires the specification of a curvature vector $\mathbf{c}$ for the following parameterization:

$$\mathbf{x}(u) = \mathbf{v}_1 + u(\mathbf{v}_2 - \mathbf{v}_1) + u(1-u)\mathbf{c}\,. \tag{2}$$

Other $2d$ surface elements include a cubic patch, an elliptic patch (a segment of an ellipse), and an arc patch (a segment of a circle).

In $3d$, patches are supported on a triangle. They are defined by a function $\mathbf{x}(u_1, u_2, u_3)$, such that

$$u_1, u_2, u_3 \geq 0 \tag{3}$$

and

$$u_1 + u_2 + u_3 = 1\,. \tag{4}$$

We use the independent variables u_1 and u_2 to describe locations on the patch surface. Thus, a location on the simplest $3d$ patch, a flat patch, is given by

$$\mathbf{x}(u_1, u_2) = \sum_{i=1}^{3} u_i \mathbf{v}_i\,, \tag{5}$$

where $u_3 = 1 - u_1 - u_2$. Here, the point $u_i = 1$ maps onto the patch vertex $\mathbf{v}_i$, and the line $u_i = 0$ maps onto the edge opposite $\mathbf{v}_i$. Adding the influence of three curvature vectors $\mathbf{c}_i$, we define a $3d$ quadratic patch as

$$\mathbf{x}(u_1, u_2) = \sum_{i=1}^{3} u_i(\mathbf{v}_i + (1-u_i)\mathbf{c}_i)\,. \tag{6}$$

Somewhat more complicated is a spherical patch, the projection onto a sphere of a triangular patch whose vertices lie on the sphere. This patch is parameterized by

$$\mathbf{x}(u_1, u_2) = \mathbf{C} + r\left(\frac{\mathbf{d}(u_1, u_2)}{|\mathbf{d}(u_1, u_2)|}\right)\,, \tag{7}$$

where $\mathbf{C}$ and r represent the sphere center and radius, respectively, and the direction vector $\mathbf{d}$ is given relative to the center as

$$\mathbf{d}(u_1, u_2) = \sum_{i=1}^{3} u_i(\mathbf{v}_i - \mathbf{C})\,. \tag{8}$$

Other $3d$ surface elements include a bicubic patch and an ellipsoidal patch (similar to the spherical

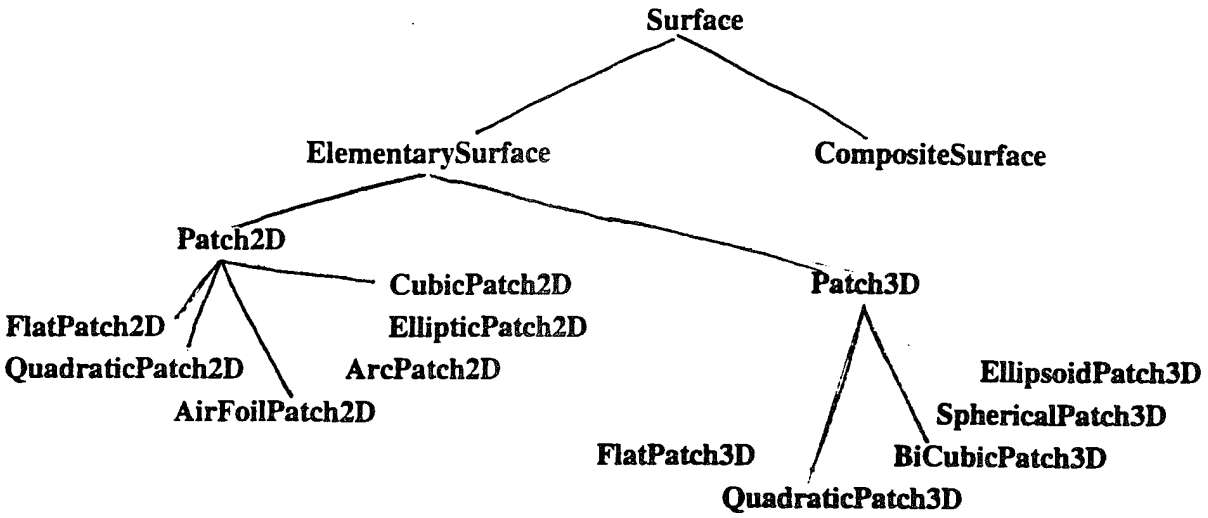

Figure 1: Surface class hierarchy

patch, but a projection onto an ellipsoid).

For FastScat, we needed code not only to implement these equations, but to provide a means of building a surface from constituent patches; computing derivatives, curvature, the metric tensor, and tangent and normal vectors at specified points; and determining parameters such as the area, extent, and spatial proximity of patches. We realized that the addition of new algorithms to FastScat might place new demands on the surface model, requiring frequent changes to the surface classes. Indeed, since one goal of our research was the effects of the surface model itself, new models would need to be added quickly and easily.

Examining some existing scattering codes, we found that assumptions about the surface model, generally flat patches, were littered throughout the code. Changing the model would require several months of work. With these issues in mind, we decided on an object-oriented design in which details of particular surface parameterizations could be fully encapsulated, and interfaces to the surface model would be clearly specified. In the next section, we discuss the design and implementation of class `Surface` and the subclasses responsible for modeling surfaces in FastScat.

3 Class Surface

For the surface code, as for the rest of FastScat, we have adopted a design strategy in which the essential elements of the problem are represented by C++ classes. Hence, in the surface class hierarchy shown in Figure 1, there are classes with names such as `Surface`, `Patch2D`, `Patch3D`, `QuadraticPatch3D`, etc.

All FastScat surface classes are derived from the abstract base class `Surface`. Though most operations allowed on a surface are defined in `Surface`, few are actually implemented there. Methods which depend on the specific surface parameterization, such as computing derivatives, must be implemented at the level where that parameterization is defined. The virtual method mechanism of C++ allows us to access such methods through interfaces defined in `Surface`, without regard

for the particular kind of Surface we are actually using. For instance, the method to translate a surface by a given vector is defined in class Surface as follows:

```
class Surface {
public:
  virtual Surface& translate(const RVector& delta) = 0;
...
}
```

The implementation of this method depends on the type of the target surface, and is discussed further in Sections 3.1.1 and 3.2. We note here, however, that defining a translate method for use on both $2d$ and $3d$ surfaces required the invention of the space vector class RVector, from which RVector2D and RVector3D are derived. The RVector argument to the translate method may thus be a vector in either the $2d$ or the $3d$ coordinate frame, as appropriate.

As shown in Figure 1, a Surface may be either an ElementarySurface, such as an individual patch, or a CompositeSurface, a collection of Surfaces (possibly other CompositeSurfaces). These two derived classes are described below.

3.1 ElementarySurface

Class ElementarySurface is an abstract base class from which the abstract classes Patch2D and Patch3D are derived. ElementarySurface is the base class of all surface elements which have a parametric representation, such as the patches described in Section 2. To aid in modeling antennas, we will eventually add wires to FastScat, creating the additional decendants Wire2D and Wire3D.

As in Surface, few methods are actually implemented in ElementarySurface. The class serves primarily as a package used to operate on parametric surface elements, without regard to their true object type. Methods are defined here which are not appropriate for the more general Surface. For example, integration over the scattering surface is ultimately done in terms of integrations over the individual elementary surfaces. At this level, we choose to work with the objects directly as members of the ElementarySurface class, where methods such as the mapping from parametric coordinates to spatial coordinates are available. Because such methods are virtual, we need not determine whether the elementary surface is $2d$ or $3d$, or a patch or a wire—the language keeps track of these details and supplies the correct implementation*.

3.1.1 Patch2D and Patch3D

As their names suggest, Patch2D and Patch3D are the base classes for $2d$ and $3d$ parametric patches, respectively. The features which distinguish a $2d$ patch from a $3d$ patch are the coordinate frame and the fact that a $2d$ patch has two vertices (or endpoints) while a $3d$ patch has three. At this point, we can begin to implement methods such as translate. In Patch3D this method simply adds the specified translation vector to each vertex as follows:

*Because specification of surface points in $2d$ requires only one parameter, whereas in $3d$ it requires two, we take care to define an interface which will work for both.

```
class Patch3D : public ElementarySurface {
public:
  virtual Surface& translate(const RVector& delta) {
    vertex1 += delta;
    vertex2 += delta;
    vertex3 += delta;
    return *this;
  }
...
private:
  RVector3D vertex1, vertex2, vertex3;
};
```

This implementation translates the coordinate data common to all types of $3d$ patches. The arithmetic operators defined in `RVector` perform type checking on the vector operands to insure that only vectors of the the same dimensionality are combined. Thus, if an `RVector2D` were mistakenly supplied to the the translate method above, the `RVector3D operator +=` would throw an exception.

The relevant data for translation of flat patches (class `FlatPatch3D`) are the vertices, so the translate method need not be implemented there — the method `Patch3D::translate` will be called by default. Translation of an object of type `SphericalPatch3D`, on the other hand, also involves moving the the center of the associated sphere. Thus, the translate method is implemented in this class as:

```
class SphericalPatch3D : public Patch3D {
public:
  Surface& translate(const RVector& delta) {
    sphere_center += delta;              // shift center
    return Patch3D::translate(delta);    // shift vertices
  }
...
private:
  RVector3D sphere_center;
};
```

Some methods are implemented in the base class (`Patch2D` or `Patch3D`) in terms of virtual functions implemented in derived classes. For example, consider the determinant of the metric tensor, g. For a patch in three dimensions its value depends on the shape of the the patch only through the partial derivatives:

$$g \equiv \det g_{ij} = \left| \frac{\partial}{\partial_{u_1}} \mathbf{x}(u_1, u_2) \times \frac{\partial}{\partial_{u_2}} \mathbf{x}(u_1, u_2) \right|^2 . \tag{9}$$

Although the derivatives themselves may need to be implemented at a lower level, g can be computed in `Patch3D`:

```
class Patch3D : public ElementarySurface {
public:
```

```
    // declare the partial derivative with respect to u1, u2
    virtual RVector3D ppu1(double u1, double u2) const;
    virtual RVector3D ppu2(double u1, double u2) const;

    // determinant of the metric tensor, g
    double g(double u1, double u2) const {
              return norm(cross(ppu1(u1,u2), ppu2(u1,u2))); }
...
}
```

The cross method performs the cross product between two vectors, and norm returns the vector norm, $x^2 + y^2 + z^2$. The partial derivative $\partial x/\partial u_1$ is implemented in Patch3D as:

```
RVector3D Patch3D::ppu1(double u1, double u2) const {
  // derivative for a flat patch
  return RVector3D(vertex1 - vertex3);
}
```

with similar code for $\partial x/\partial u_2$. This code is exactly that needed for flat patches, so the class FlatPatch3D does not have its own implementation. The class QuadraticPatch3D does, however, need new implementations for ppu1 and ppu2. For ppu1 this is:

```
RVector3D QuadraticPatch3D::ppu1(double u1, double u2) const {
  return Patch3D::ppu1(u1,u2) + (1 - 2*u1)*C1 - (1 - 2*u1)*C3;
}
```

Though the methods we have shown here seem simple, they are quite typical of code found in the patch classes. Many patches we have implemented, such as the BiCubicPatch3D, have much more complicated parameterizations than those we have discussed. They are, however, no more difficult to add, since implementing each new patch type is really just a matter of plugging the appropriate equations into the framework supplied by the design.

3.2 CompositeSurface

The class which represents a collection of Surfaces is named CompositeSurface. It is a list of Surfaces, where most of the abstract methods inherited from Surface are implemented by calling the appropriate routines for the constituent surface pieces through the virtual method mechanism of C++.

With the use of the utility class SurfaceIterator, whose purpose is to iterate over the components of a CompositeSurface, the translation method discussed previously is implemented as follows:

```
Surface& CompositeSurface::translate(const RVector& delta) {
  for (SurfaceIterator si(*this); !si.end(); si++)
    // calls the virtual "translate" method for each surface
    ((Surface&) si).translate(delta);
  return *this;
}
```

If the CompositeSurface contains other CompositeSurfaces, this method is called recursively, eventually reaching the ElementarySurfaces. Implementation of the methods for scaling and rotation are similar.

An important method of CompositeSurface appends a Surface to a CompositeSurface. To do this it needs to make a deep copy of the surface elements to be added. Unfortunately, the types of these particular elements are not known to the user, and C++ constructors cannot be made virtual. This lead us to defining a duplicate method which acts like a simulated virtual constructor. As usual, the method is defined in Surface:

```
class Surface {
public:
  virtual Surface* duplicate() = 0;
...
}
```

It is implemented in the derived classes by calling the class's constructor, i.e.:

```
class FlatPatch3D : public Surface {
public:
  virtual Surface* duplicate() const {
    return new FlatPatch3D(*this);
  }
...
}
```

Then, the code:

```
void CompositeSurface::append(const Surface& s) const {
  List::append(s.duplicate());
}
```

duplicates the specified surface and appends it onto the surface list.

These examples show how, with the use of polymorphism, CompositeSurface allows the manipulation of groups of surface elements as a single surface entity. This feature makes possible the succinct expression of many operations, such as the reading and writing of complete surfaces, and allows us to form surfaces made of a wide variety of patch types.

4 Notes on Use and Efficiency

The surface class hierarchy described here has been very important to the development and maintainace of FastScat. Because the design parallels the way we think about surfaces in problems of scattering, we have found the resulting code to be more readable and less prone to error than code we have written for similar purposes in Fortran, for instance. Manipulation and queries of surfaces are expressed in a natural and intuitive way. Once a workable and efficient parametric description of a patch is determined, the addition of a new surface type is relatively easy. It involves only the

creation of a new decendent of the appropriate patch base class. and the implementation of the required methods (about 30 as of this writing).

Adding new surface "features" is no more difficult. This year we plan to add material characteristics to patches in FastScat to model dielectrics. This will require the creation of new data members and the implemention of related methods in class `ElementarySurface`. Occasionally, we need to add functions to the surface class which depend directly on the surface parameterization. In this case, new methods may need to be added at the lowest levels of the surface hierarchy. In any case, the rest of FastScat is unchanged.

Efficiency has been of particular concern to us in the development of FastScat. Our surface classes are used primarily to solve integral equations using high order quadrature rules, which require functions to be evaluated at many points on each patch. Often the partial derivatives, surface normals, metric tensor, curvature, and other quantities are evaluated at those same points. To minimize function calls, and yet to preserve the polymorphism of our classes, we pass arrays of data to such methods, making only a single virtual call to each method for all the points on a particular patch. These methods, in turn, return arrays of results which can be manipulated further. The computation of the determinant of the metric tensor (discussed previously in Section 3.1) is a good candidate for this sort of technique. The relevant code can be rewritten as follows:

```
class Patch3D : public ElementarySurface {
public:
  // declare the partial derivative with respect to u1, u2
  virtual RV3DArray ppu1(const RArray u1u2) const;
  virtual RV3DArray ppu2(const RArray u1u2) const;

  // determinant of the metric tensor, g
  RArray g(const RArray u1u2) const {
            return norm(cross(ppu1(u1u2), ppu2(u1u2))); }
...
}

RV3DArray Patch3D::ppu1(const RArray u1u2) const {
  // return the number of coordiantes in the array
  int N = u1u2.length()/2;
  // compute derivative at N points
  return RV3DArray(N, vertex1 - vertex3);
}

RV3DArray QuadraticPatch3D::ppu1(const RArray u1u2) const {
  int N = u1u2.length()/2;
  // compute derivative at N points
  RV3DArray ppu1A(Patch3D::ppu1(u1u2));
  for (int n = 1; n <= N; n++) {
    int index = n*2;
    double u1 = u1u2(index-1);
    double u2 = u1u2(index);
    ppu1A(n) += (1 - 2*u1)*C1 - (1 - 2*u1)*C3;
```

```
    }
    return ppu1A;
  }
```

In the above implementation, we compute g for several points at once. Thus, an array of points (`RArray`) is passed to the derivative functions, and an array of vectors is returned (`RV3DArray`). The `cross` method performs the cross product between corresponding vectors in arrays of vectors, and `norm` returns an array of the vector norms, $x^2 + y^2 + z^2$. This technique resulted in a factor of five speed-up over the "one point at a time" strategy.

5 Concluding Remarks

We have developed a general purpose surface class library for the representation of complex, continuous and discontinuous surfaces made up of collections of parametric patches. We have found that a robust program design requires a good understanding of the problem, and of the possible future requirements of the implementation. The surface classes described are the result of several design iterations. In particular, our first attempts were very inefficient and the program spent most of its time doing virtual function calls instead of floating point arithmetic. Fortunately, having followed the rules of object-oriented programing, most of the redesigned code was sufficiently isolated that damage to the rest of the program was minimal.

In the early stages of this work, many recommended against our use of an object-oriented design and C++ for FastScat. Although, from time to time, we have had to work around problems related to immature development tools, compiler bugs, and language elements not being implemented, we are confident that object-oriented methods have been beneficial. In 1993, FastScat excelled in speed and accuracy in a competion at the Second International Workshop on Approximations and Numerical Methods for the Solution of the Maxwell Equations. Although our success is primarily a result of our advanced numerical techniques, these would have been difficult, if not impossible, to implement in this much generality without the use of object-oriented programming techniques.

Acknowledgements The authors would like to thank Stephen Wandzura for useful discussions of surface parameterization and his guidance in the determination of our long-term implementation goals.

References

[CRW93] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. The fast multipole method: A pedestrian prescription. *IEEE Antennas and Propagation Society Magazine*, 35(3):7–12, June 1993.

[HRS+93] Lisa Hamilton, Vladimir Rokhlin, Mark Stalzer, R. Steven Turley, John Visher, and Stephen Wandzura. The importance of accurate surface models in RCS computations. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 3, pages 1136–1139, Ann Arbor, MI, June 1993. IEEE.

[HST+93a] Lisa Hamilton, Mark Stalzer, R. Steven Turley, John Visher, and Stephen Wandzura. Scattering computation using the fast multipole method. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 2, pages 852–855, Ann Arbor, MI, June 1993. IEEE.

[HST+93b] Lisa Hamilton, Mark Stalzer, R. Steven Turley, John Visher, and Stephen Wandzura. Method of moments scattering computations using high-order basis functions. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 3, pages 1132–1135, Ann Arbor, MI, June 1993. IEEE.

[HST+93c] Lisa Hamilton, Mark Stalzer, Steve Turley, John Visher, and Stephen Wandzura. FastScat: an object-oriented program for fast scattering computation. In *Proceedings of the First Annual Object-Oriented Numerics Conference*, pages 247–256, Corvallis, OR, 1993. Rogue Wave Software.

THE FAST MULTIPOLE METHOD FOR PERIODIC STRUCTURES[†]

Vladimir Rokhlin
Fast Mathematical Algorithms and Hardware Corporation
Hamden, CT 06514

Stephen Wandzura*
Hughes Research Labs
Malibu, CA 90265

Abstract

The fast Multipole method (FMM) can be applied to the computation of radiation and scattering from infinite periodic structures by applying the Ewald summation technique to the FMM translation operator. When the FMM group is taken to be a unit cell in the structure, this results in a large saving in the setup times compared to a straightforward FFT approach.

1 Introduction

The fast Fourier transform (FFT) can be used to accelerate the computation of radiation or scattering from an infinte periodic structure[1]. For many problem dimensions of interest, the computation of Galerkin matrix elements of the Helmholtz operator dominates the calculation time. This is because of the large number of evaluations of the periodic free-space Green functions required to do accurate quadratures. The fact that most of the required computation represents far interactions suggests that the fast Multipole method[2] (FMM) might be used to ameliorate the problem. In this paper, we show how this can be done; the two basic parts of the calculation being the the splitting of the interactions into near and far parts, and the application of the Ewald summation technique to the Hankel functions that appear in the FMM translation operator. If the FMM group taken to be a unit cell (which is optimal, unless the unit cell is more than several wavelengths in extent), no savings in the application of the impedance matrix results, but the setup time is vastly reduced. In essence, the FMM simply provides an efficient mechanism for computation of the matrix elements of the periodic Green function, which can then be solved in the conventional way.

2 Details

For clarity, we analyse the technique for $1d$ arrays only, the extension to $2d$ being straightforward. We consider an infinite array that has a current that is represented by Galerkin expansion coefficients $J_{\alpha,\beta}$, where α labels the unknowns on a single

[†]This research was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract Numbers F49620-91-C-0064 and F49620-91-C-0084. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereron.

element and β labels the element within the array. The periodicity of the structure is reflected in the translational symmetry of the impedance matrix,

$$Z_{\alpha\beta\alpha'\beta'} = Z_{\alpha\alpha'}(\beta - \beta') . \tag{1}$$

The current is assumed to be periodic with "supercell" period N:

$$J_{\alpha,\beta} = J_{\alpha,\beta+N} . \tag{2}$$

The spatial displacement induced by a unit increment of β is denoted S. The translational symmetry of the impedance matrix and the current enables one to rapidly apply the matrix by discrete Fourier transformation in β, so that application requires $\mathcal{O}(NU^2)$ operations, where U is the number of unknowns per unit cell[1]. One may obtain an arbitrarily accurate approximation of the response of an infinite array to any excitation by using a sufficiently large value of N.

2.1 Separation of the free-space Green function

Since the factorization of the FMM can only be applied to interactions that are sufficiently far, we split the impedance matrix into near and far parts:

$$Z = Z' + M , \tag{3}$$

where $Z'(\beta - \beta') = 0$ for $k|\beta - \beta'|S > L$, $M(\beta - \beta') = 0$ for $k|\beta - \beta'|S <= L$, k is the free space wavenumber and and L is the number of terms kept in the summation of the FMM translation operator.

The advantage of the near-interaction matrix Z' is that no sums are needed in its computation to represent the (supercell) periodicity. Its computation is thus exactly like that for arbitrary structures.

The advantage of the M is that since it may be factorized by the FMM method, the sums representing the supercell periodicity may be confined to the FMM translation operator T, resulting in far fewer time consuming Ewald summations. After construction of the FMM factorization, the matrix elements of $M_{\alpha\alpha'}(\beta - \beta')$ may be computed by matrix multiplication, added to those of Z', and Fourier transformed to obtain[1] $\mathcal{Z}_{\alpha\alpha'}(q)$, where q is the Fourier transform variable conjugate to $\beta - \beta'$.

2.2 Ewald summation technique for the FMM translation operator

The translation operator for the periodic structure

$$T(\beta - \beta', \hat{k}) = \frac{k}{(4\pi)^2} \sum_{\gamma=-\infty}^{\infty} \sum_{l=0}^{L} i^l (2l+1) h_l^{(1)}(kX_{\beta,\beta'+N\gamma}) P_l(\hat{k} \cdot \hat{X}_{\beta,\beta'+N\gamma}) , \tag{4}$$

were $\mathbf{X}_{\beta\beta'} = (\beta - \beta')\mathbf{S}$, and unit vectors are denoted as $\hat{x} \equiv \mathbf{x}/x$. The Ewald summation technique[3, 4] works by splitting the potential

$$\Phi_l(\mathbf{x}) = P_l(\hat{k} \cdot \hat{x}) h_l^{(1)}(kx) \tag{5}$$

into two pieces, one that decays rapidly in space, and the other that has a rapidly convergent Fourier transform:

$$\Phi_l(\mathbf{x}) = \hat{\Phi}_l(\mathbf{x}) + \tilde{\Phi}_l(\mathbf{x}) \tag{6}$$

$$\hat{\Phi}_l(\mathbf{x}) = \sqrt{\frac{2}{\pi}}\frac{(2a)^{l+3/2}}{k^l} e^{k^2/4a} P_l(\hat{k}\cdot\hat{x}) \int_x^\infty dt\, t^{l+2} e^{-at^2} \left[h_l^{(1)}(kr) j_l(kt) - j_l(kr) h_l^{(1)}(kt)\right] \tag{7}$$

$$\tilde{\Phi}_l(\mathbf{x}) = \frac{2}{i\pi k^{l+1}} P_l(\hat{k}\cdot\hat{x}) \int_0^\infty dp \frac{p^{l+2} e^{-(p^2-k^2)/4a} j_l(pr)}{p^2 - k^2 - i\epsilon}, \tag{8}$$

where a is a constant chosen to optimize the computation. The $\hat{\Phi}(\mathbf{x})$ part may be summed directly (along with the subtraction of any discrete parts that are included with Z') because the sum may be truncated (at distance $\propto 1/\sqrt{a}$) with arbitrarily small error. The indefinite integral in $\hat{\Phi}(\mathbf{x})$ is a function of one variable (x), and so may be tabulated and interpolated. The sum over the $\tilde{\Phi}$ part can be done using the identity

$$\sum_{\gamma=-\infty}^{\infty} \tilde{\Phi}_l(\mathbf{x}+\gamma\mathbf{S}) = \frac{1}{S}\sum_{\mu=-\infty}^{\infty} \int_{-\infty}^{\infty} dx' \exp\frac{2\pi i\mu x'}{S} \tilde{\Phi}(\mathbf{x}+x'\hat{S}) \tag{9}$$

$$= \frac{2i^{l-1}}{k^{l+1}S} P_l(\hat{k}\cdot\hat{S}) \sum_{\mu=-\infty}^{\infty} \exp\left(-\frac{2\pi i\mu x}{S}\right) \int_{2\pi\mu/S}^{\infty} dp\, p^{l+1} P_l\left(\frac{2\pi\mu}{pS}\right) \frac{e^{-(p^2-k^2)/4a}}{p^2-k^2-i\epsilon}, \tag{10}$$

where the sum over μ may be truncated with arbitrarily small error. (The dx' integration is considerably simplified because $\mathbf{x}$ and $\mathbf{S}$ are parallel.) The integrations over p may be tabulated as a function of μ and can be done numerically using

$$\int_0^\infty dp \frac{f(p^2)}{p^2-k^2-i\epsilon} = \frac{i\pi}{2k} f(k^2) + \int_0^\infty dp \frac{f(p^2)-f(k^2)}{p^2-k^2}. \tag{11}$$

References

[1] Winifred Kummer, R. Steven Turley, and Stephen Wandzura. Application of the fast Fourier transform to infinite periodic structures. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 3, pages 1246–1249, Ann Arbor, MI, June 1993. IEEE.

[2] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. The fast multipole method: A pedestrian prescription. *IEEE Antennas and Propagation Society Magazine*, 35(3):7–12, June 1993.

[3] J. M. Ziman. *Principles of the Theory of Solids.* Cambridge University Press, Cambridge, second edition, 1972.

[4] K.E. Jordan, G.R. Richter, and P. Sheng. An efficient numerical evaluation of the Green's function for the Helmholtz operator on periodic structures. *J.Comp.Phys*, 63:222–235, 1986.

Appendix B

Second International Conference and Workshop on Approximations & Numerical Methods for the Solution of the Maxwell Equations

FastScat[1], Hughes Research Laboratories

L.R. Hamilton, P.A. Macdonald, M.A. Stalzer, R.S. Turley,
J.L. Visher, S.M. Wandzura

INTRODUCTION

We ran the problems for this symposium using the FastScat program developed at Hughes Research Laboratories. The current version of this program computes electromagnetic scattering from arbitrary conducting shapes in 2 and 3 dimensions.

The computations in FastScat are based on the application of the method of moments to the (frequency domain) electric field integral equation (Rao, Wilton & Glisson 1982). We have taken advantange of recent research demonstrating the importance of using accurate surface descriptions (Hamilton et al. 1993), high order basis functions (Hamilton et al. 1993a), and careful quadratures (Ma, Rokhlin & Wandzura 1993).

All of the problems were computed with exact analytical surfaces, eliminating any errors resulting from inadequate geometry descriptions. This eliminated errors introduced by approximating curved surfaces with line segments or piecewise continuous polynomials as is often done.

We have found that the use of high order functions in the expansion of the surface current saves both memory and CPU time over low order basis and testing functions such as δ-functions, pulses, or "rooftop" basis functions. We use the Galerkin technique because of decreased sensitivity of far field quantities to errors in the surface current (Wandzura 1991).

An important feature of FastScat permits users to choose an appropriate trade-off between computation speed and accuracy. We were thus able to adjust patch sizes, basis function orders, and quadrature orders to get reliable estimates of the accuracy of our computed cross sections. Our use of high order methods permits achieving extra digits of precision with only modest increases in required computer time and memory.

BASIS FUNCTIONS

For 2d scattering, the surface elements (patches) in FastScat may be flat or curved segments which are parameterized by a function $\vec{x}(u)$, $0 \leq u \leq 1$. In the 2d Dirichlet

[1] FastScat is a trademark of Hughes Aircraft Company.

problem (TM), we use an orthonormal basis set in which the n^{th} order member is given by

$$f_n(u) = \frac{\sqrt{2n+1}}{\sqrt[4]{g(u)}} P_n(2u-1)\,, \tag{1}$$

where P_n are the Legendre polynomials, and $g(u) = |d\vec{x}/du|^2$. Using orthonormal functions generally provides for the best conditioned impedance matrix, and this choice makes the iterative solution to the MoM equations invariant under orthogonal transformations.

The integral equation for the Neumann (TE) boundary condition requires computation of the tangential derivative of the basis functions, which must therefore be continuous across patch boundaries. In this case, we use a "rooftop" function, which is non-zero and continuous over pairs of adjacent patches, as the lowest order member of our set. For every two touching patches, p^+ and p^-, we define the basis function as the sum of two functions, f^+ and f^-, each non-zero on only a single patch. Assuming that p^+ and p^- share their endpoint corresponding to $u = 1$ and $u = 0$, respectively, (i.e. $p^+(1) = p^-(0)$) we write

$$f_n^+(u) = \left(\sqrt[8]{\frac{g^+(1)}{g^-(0)}}\right) \frac{u}{\sqrt[4]{g^+(u)}} \tag{2}$$

$$f_n^-(u) = \left(\sqrt[8]{\frac{g^-(0)}{g^+(1)}}\right) \frac{1-u}{\sqrt[4]{g^-(u)}}\,. \tag{3}$$

Thus, both functions are non-zero at the shared endpoint and then decrease to zero at the other. The constant multipliers in Equations (2) and (3) make the functions continuous at the shared patch boundary. We complete the set with a series of higher order polynomials defined over individual patches which vanish at both endpoints:

$$f_n(u) = \frac{P_{n+2}(2u-1) - P_n(2u-1)}{\sqrt[4]{g(u)}}\,, \quad n \geq 1\,. \tag{4}$$

The basis functions in the set are almost all orthogonal.

RESULTS

In the following sections, we present the results for problems E.1, E.3, E.4, E.6, O.1, A.1, A.3, A.5, A.7, C.1, C.2, C.3. In each case we have chosen the bistatic angle so that forward scattering is an angle of 180° and monostatic scattering (backscattering) is an angle of 0°.

FastScat is capable of computing RCS with selectable accuracy. We present results accurate to the plot resolution. Superimposed on the standard carteisian plot in workshop format, we have included graphs of the cross section error for compuations with about 2 significant digits. The error was computed by taking the results with 8 significant digits to be exact.

ELLIPSES

The ellipses were segmented using elliptical sections. Let the equation of a point on the ellipse, (x, y), be given by

$$x = a\cos(u) \tag{4}$$
$$y = b\sin(u)\,; \tag{5}$$

where a and b are the lengths of the semi-major and semi-minor axes and u is a parametric coordinate which ranges from 0 to π. Each segment of the ellipse had an equal range in u, causing the patches near the two narrow ends to be slightly shorter than those in the flatter regions. We used 8 patches for problems E.1 and E.3 and 40 patches for the larger ellipses in problems E.4 and E.6. A 20 patch discretization we used for problems E.4 and E.6 is shown in Figure 1.

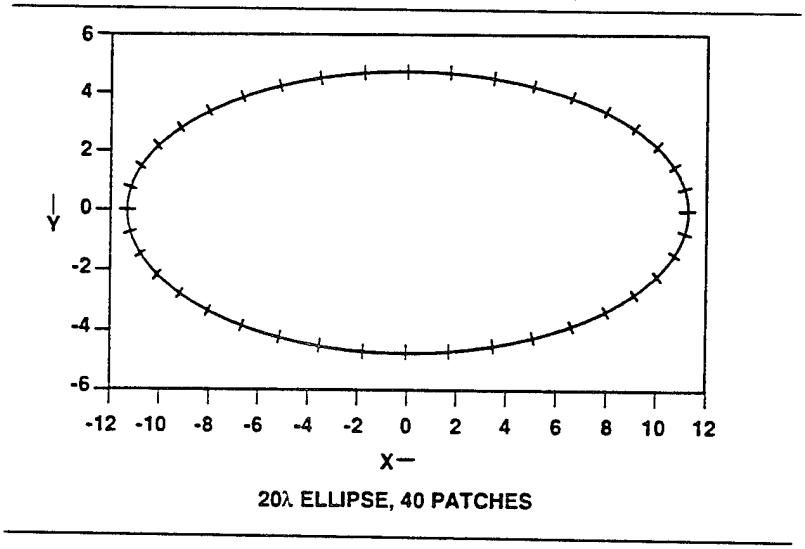

Figure 1 Sample discretization for problems E.4 and E.6.

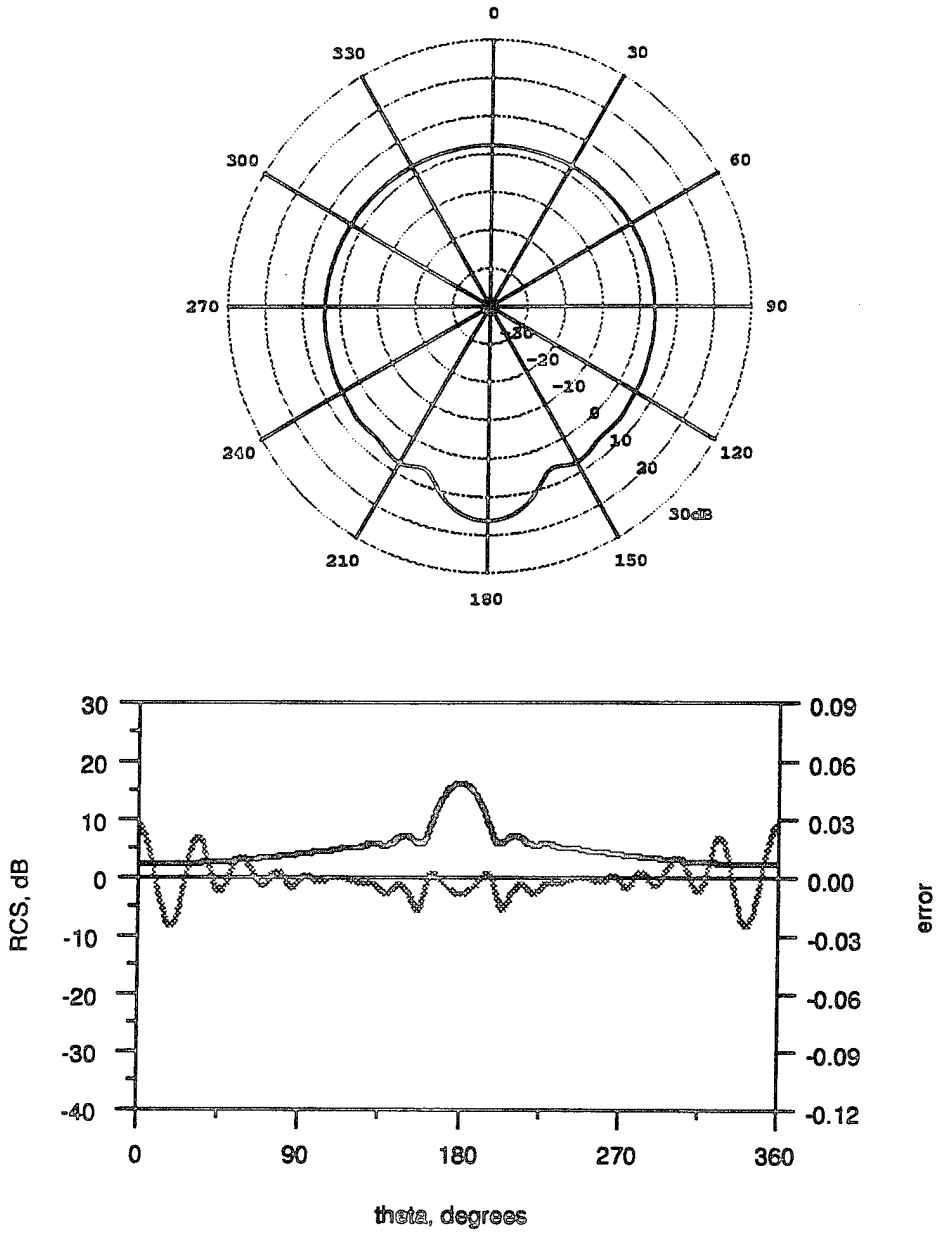

Figure 2 Cross Section for TM Scattering for Problem E.1. The light curve in the Cartesian plot is the estimated error in the result.

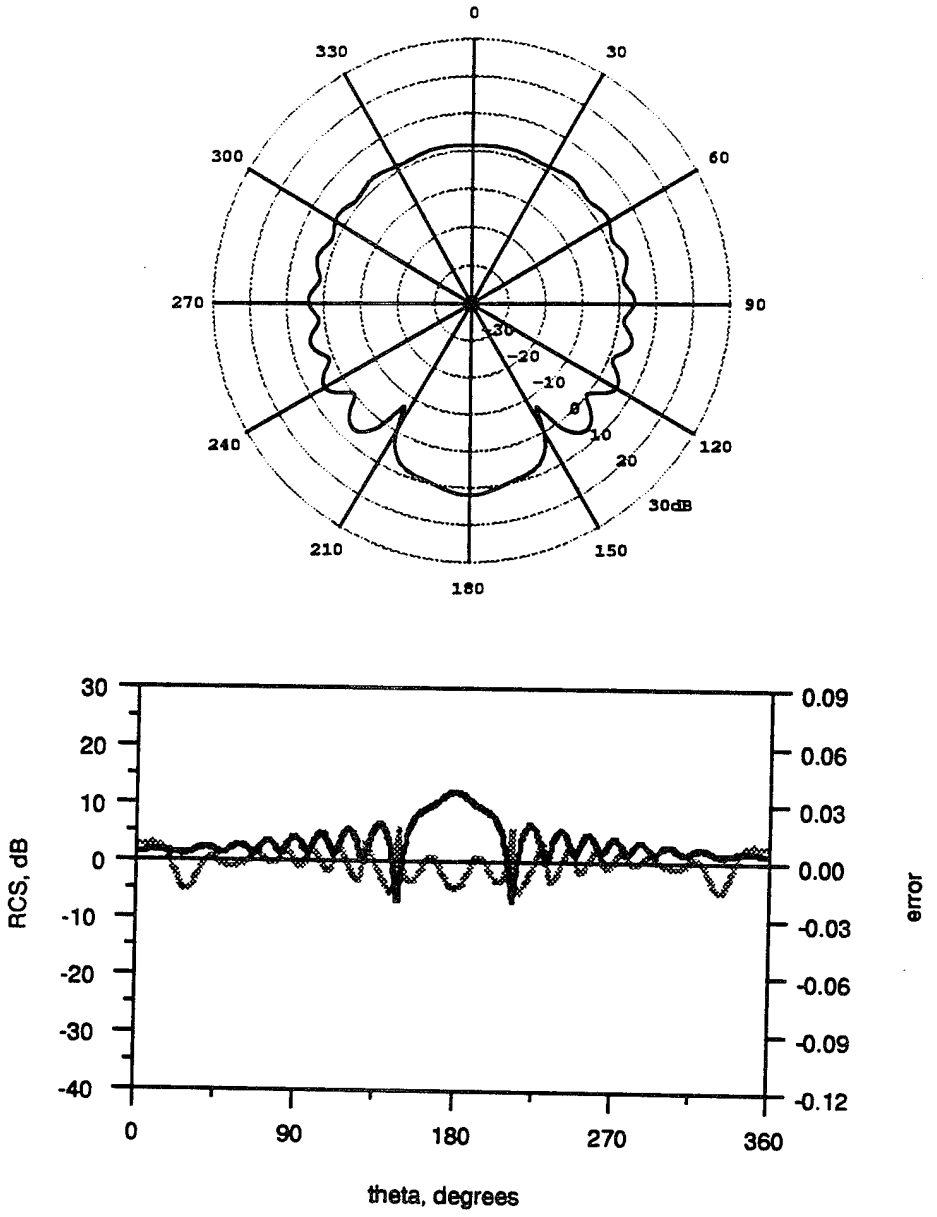

Figure 3 Cross Section for TE Scattering for Problem E.1. The light curve in the Cartesian plot is the estimated error in the result.

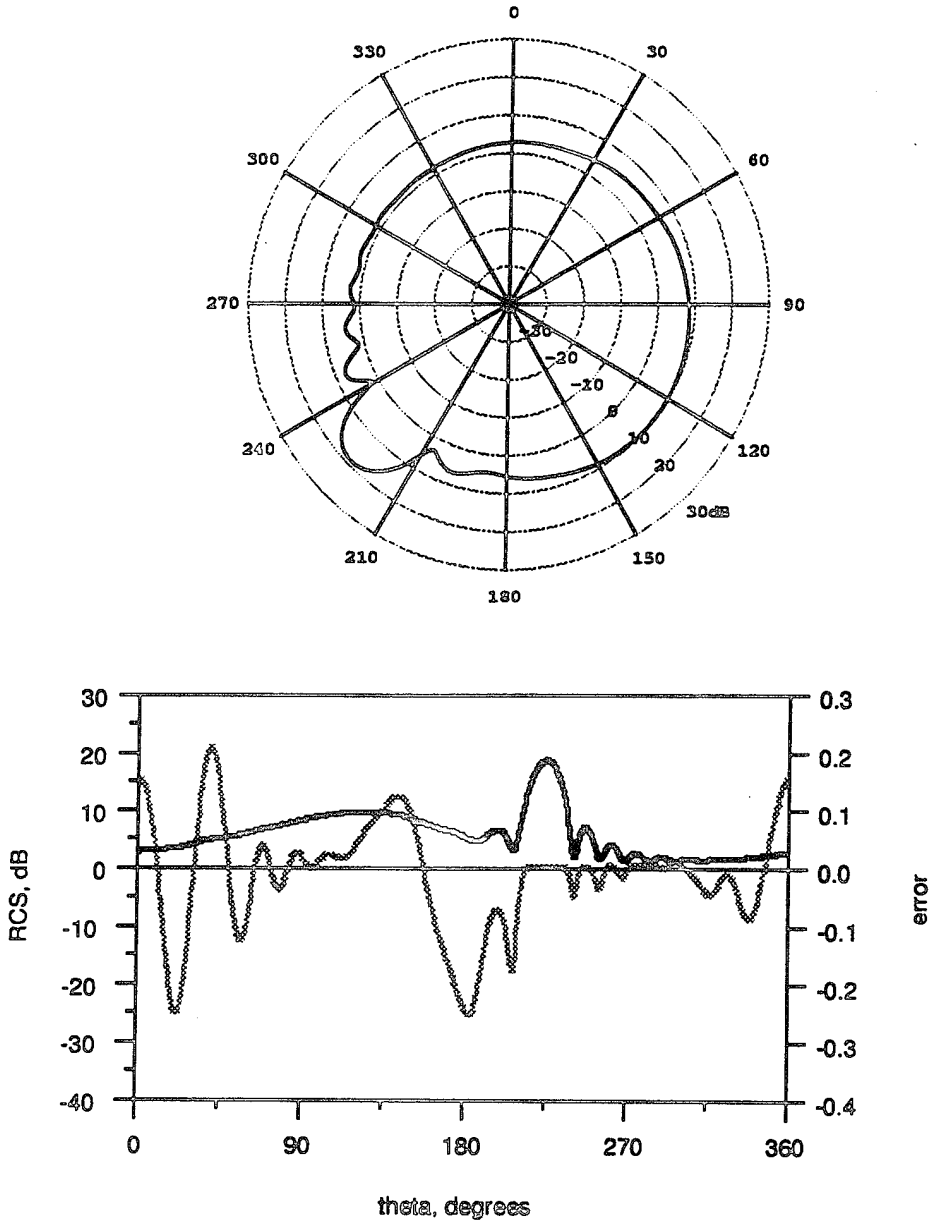

Figure 4 Cross Section for TM Scattering for Problem E.3. The light curve in the Cartesian plot is the estimated error in the result.

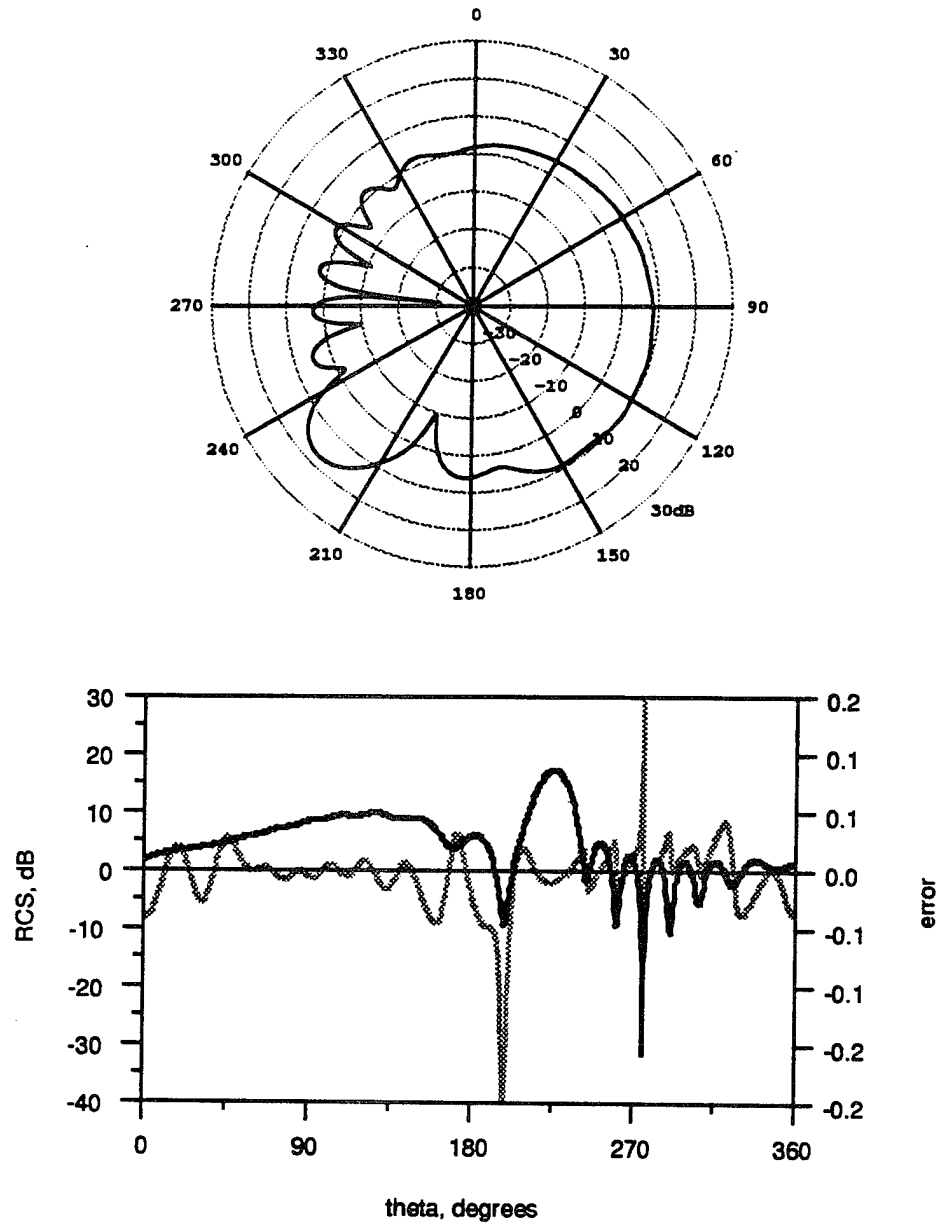

Figure 5 Cross Section for TE Scattering for Problem E.3. The light curve in the Cartesian plot is the estimated error in the result.

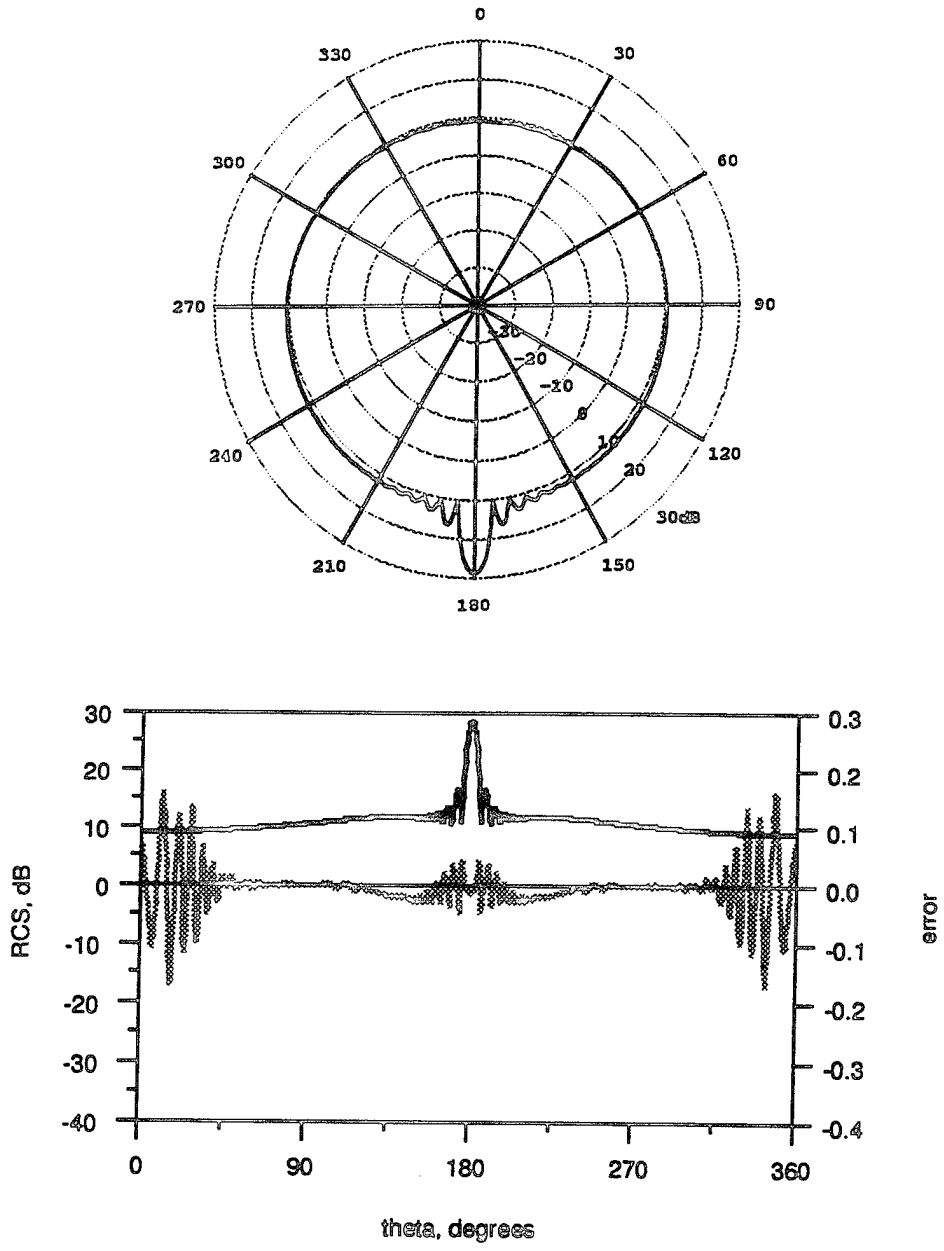

Figure 6 Cross Section for TM Scattering for Problem E.4. The light curve in the Cartesian plot is the estimated error in the result.

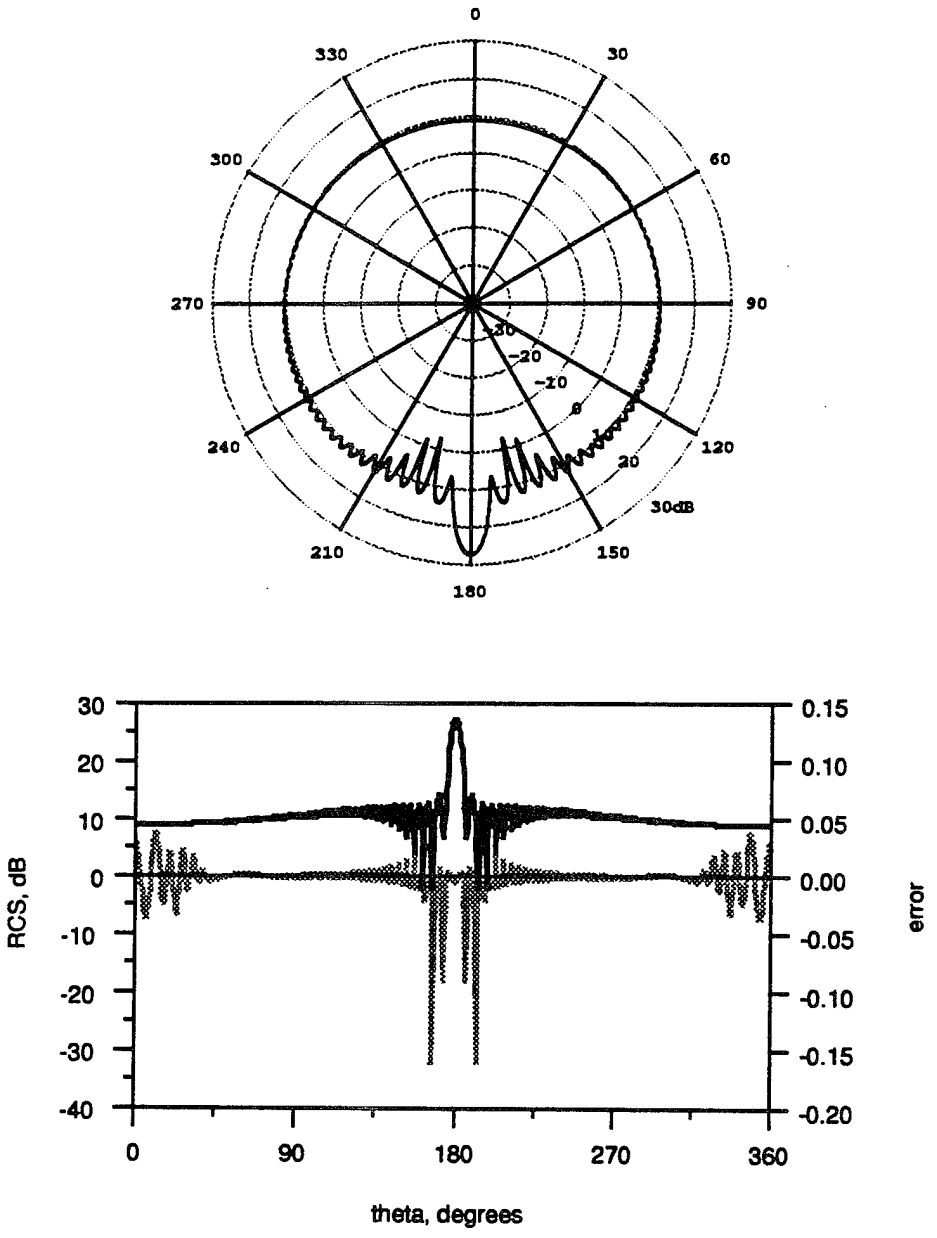

Figure 7 Cross Section for TE Scattering for Problem E.4. The light curve in the Cartesian plot is the estimated error in the result.

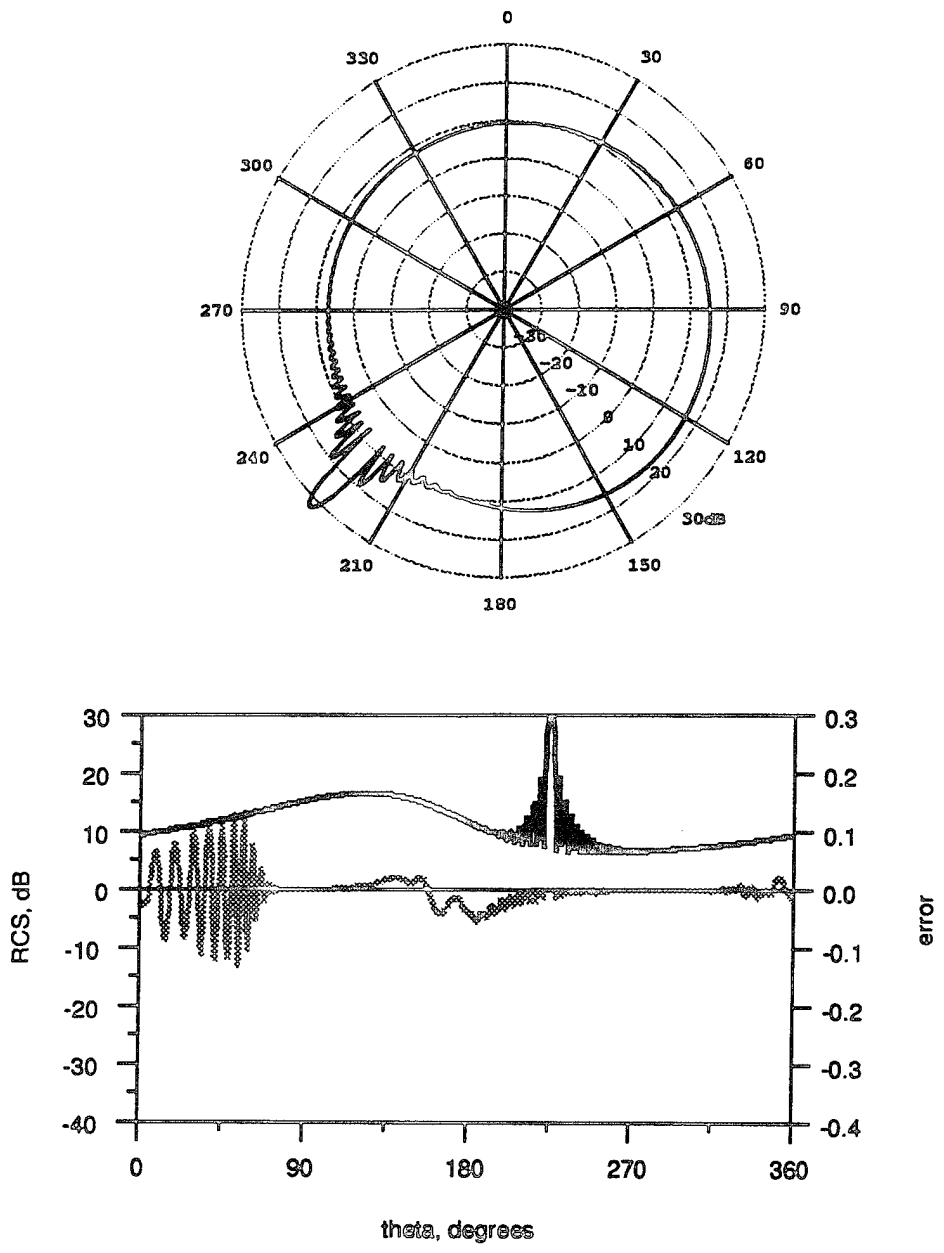

Figure 8 Cross Section for TM Scattering for Problem E.5. The light curve in the Cartesian plot is the estimated error in the result.

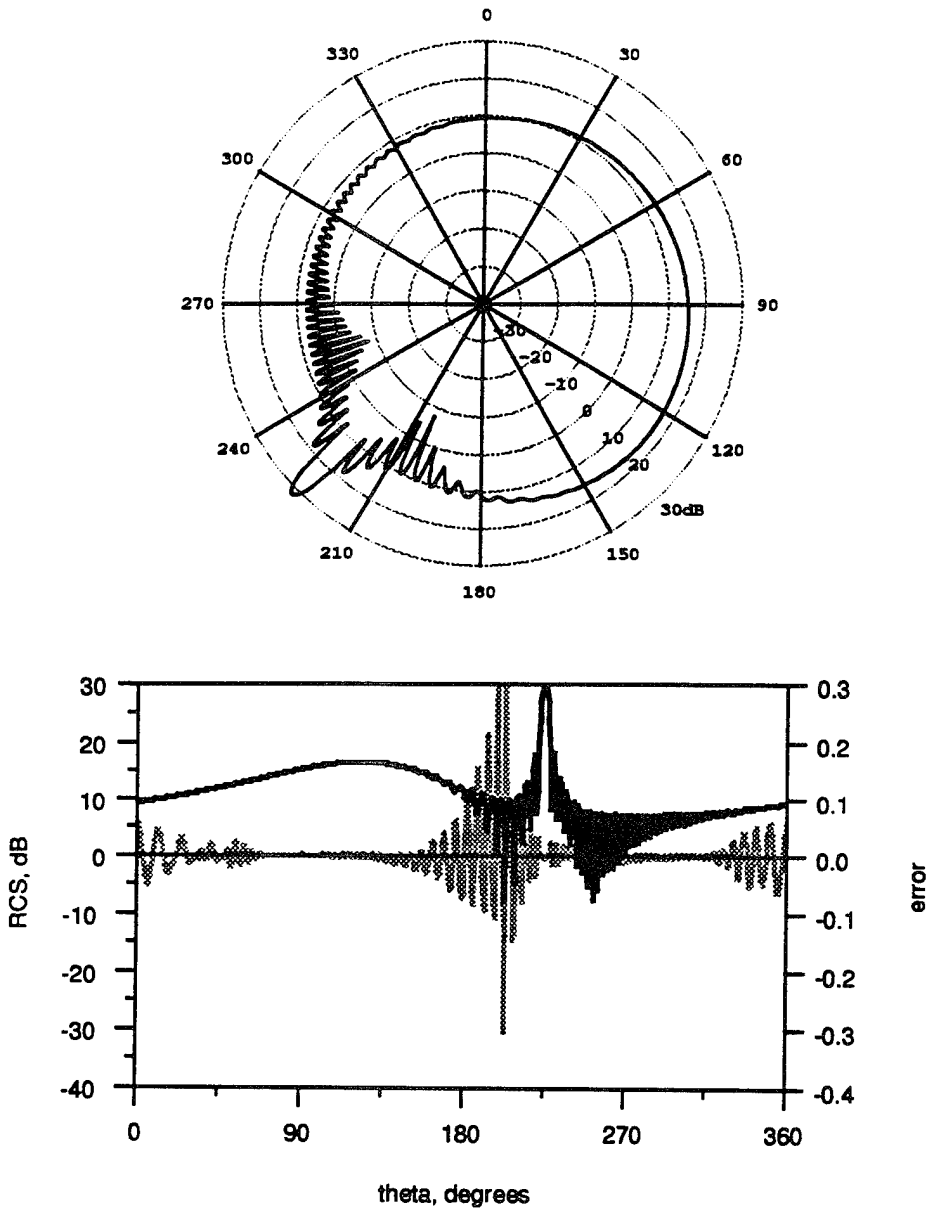

Figure 9 Cross Section for TE Scattering for Problem E.6. The light curve in the Cartesian plot is the estimated error in the result.

OGIVE

The ogive was patched using six arcs of equal length. The four patches at the tips were then subdivided a number of times to permit better modeling of the current singularities there. In Figure 10, we show an example of the ogive patching when each of the end segments was tapered 3 times.

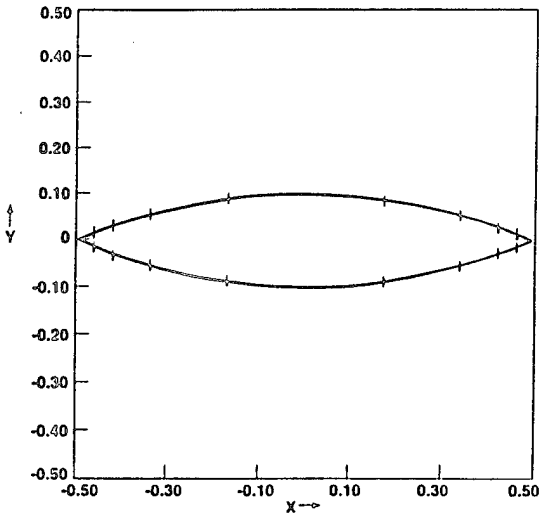

Figure 10 Sample discretization for problem O.1.

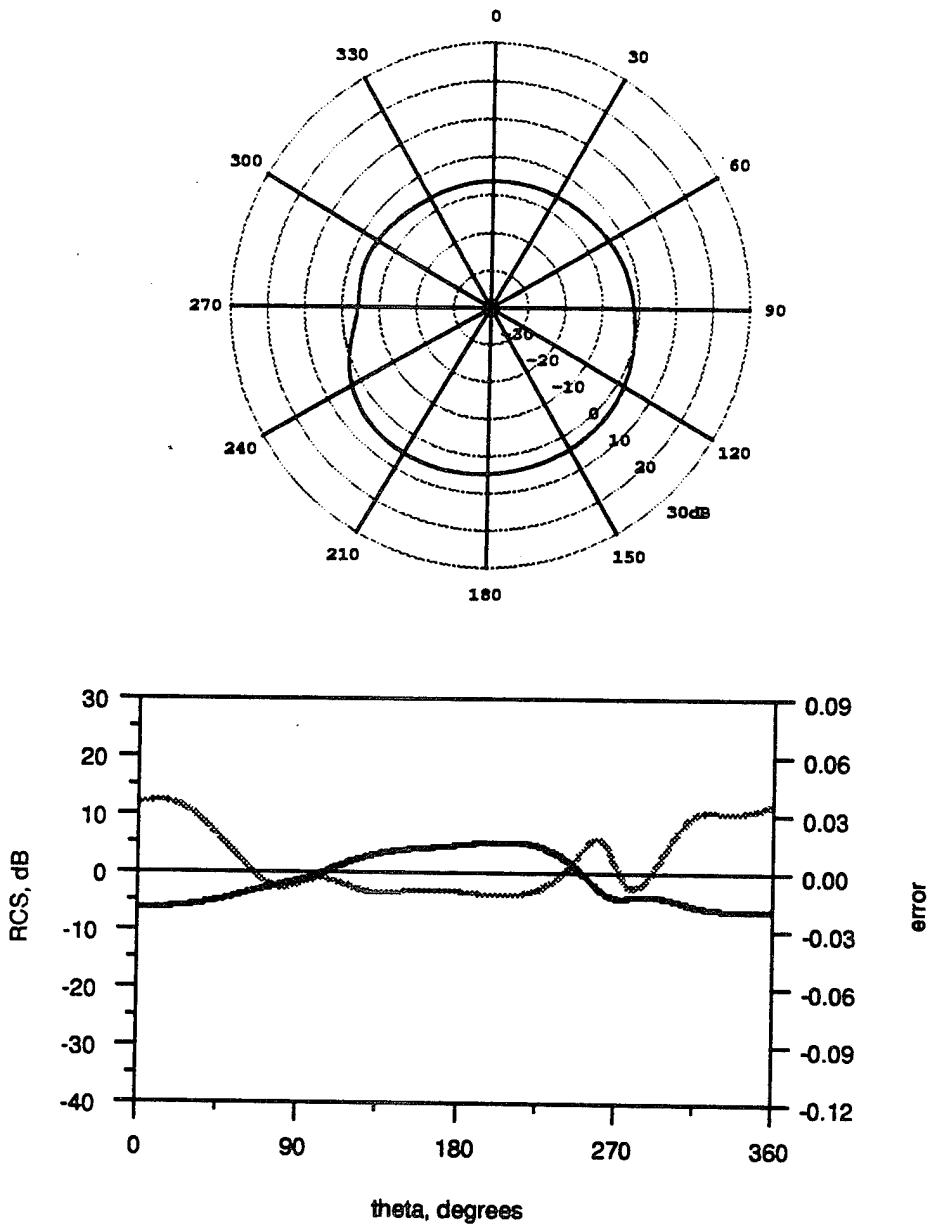

Figure 11 Cross Section for TM Scattering for Problem O.1. The light curve in the Cartesian plot is the estimated error in the result.

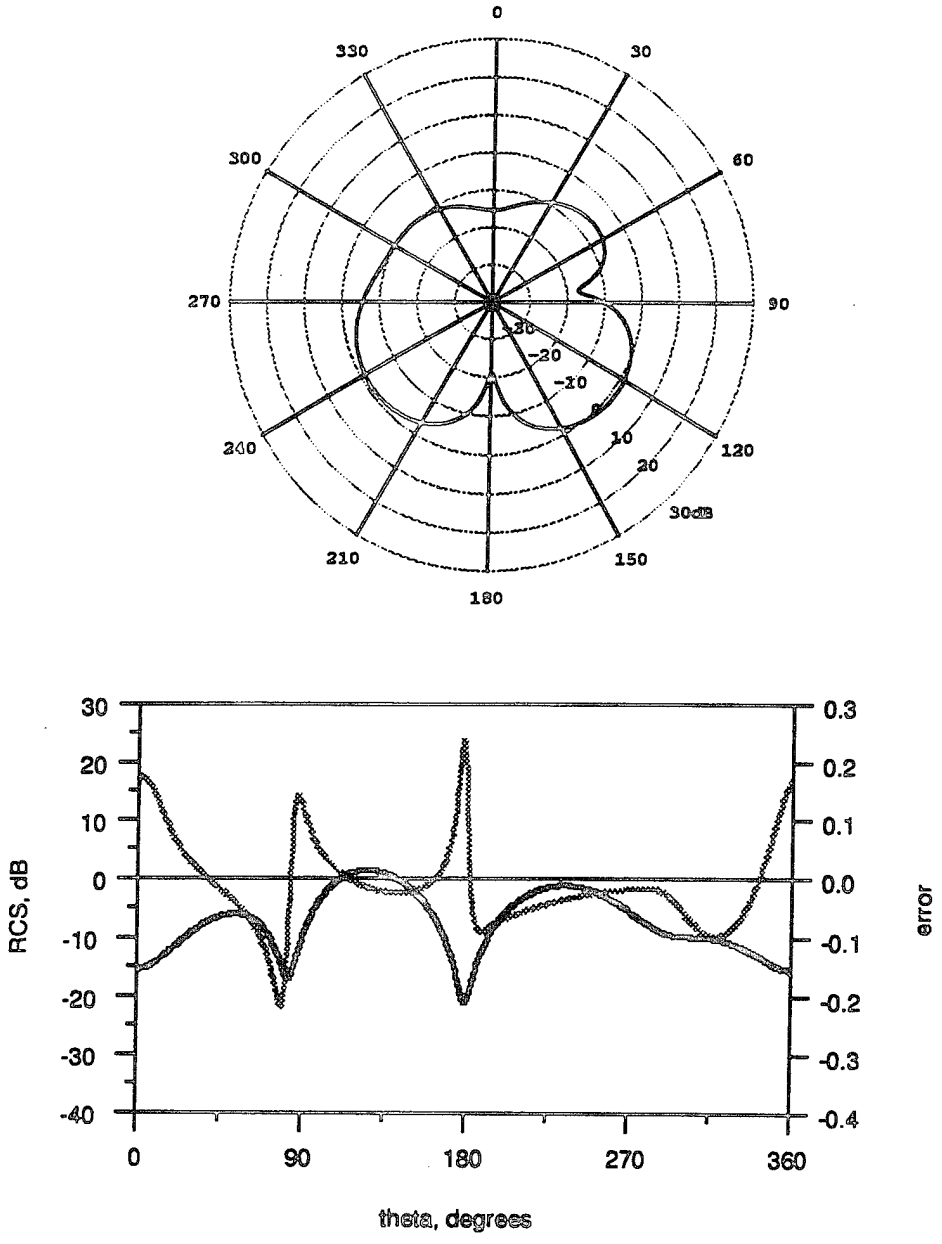

Figure 12 Cross Section for TE Scattering for Problem O.1. The light curve in the Cartesian plot is the estimated error in the result.

AIRFOILS

The patch sizes on the airfoil were chosen similarly to those for the ogive. Equal length patches were used along the surface except near the tips. At the tips the patches were tapered to better model the current singularity there. Figures 13 and 14 show examples of how the single and double airfoils were discretized.

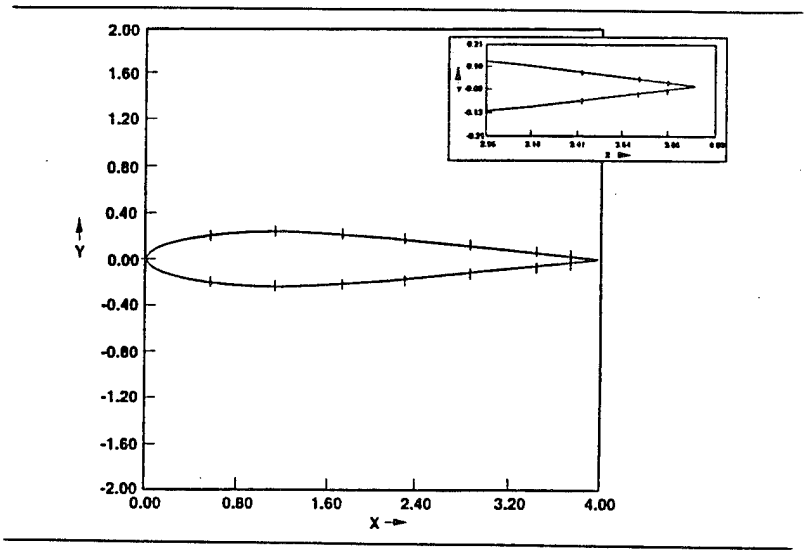

Figure 13 Sample discretization for problems A.1 and A.3.

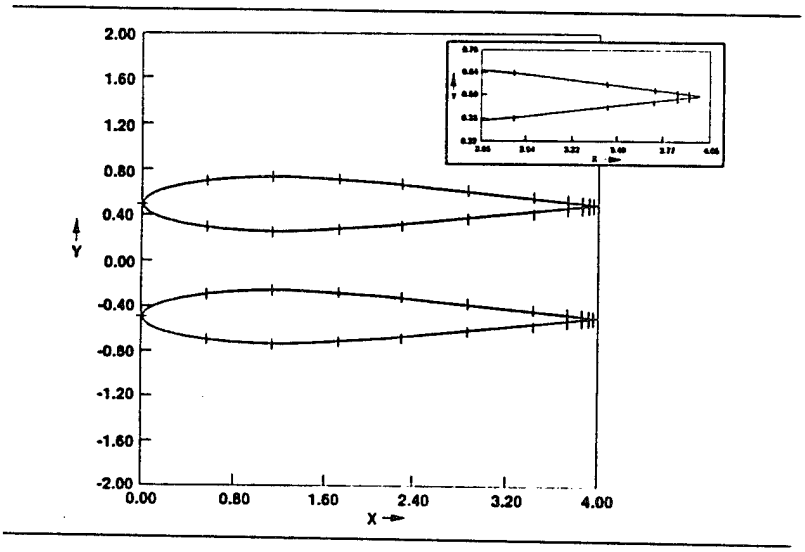

Figure 14 Sample discretization for problems A.5 and A.7.

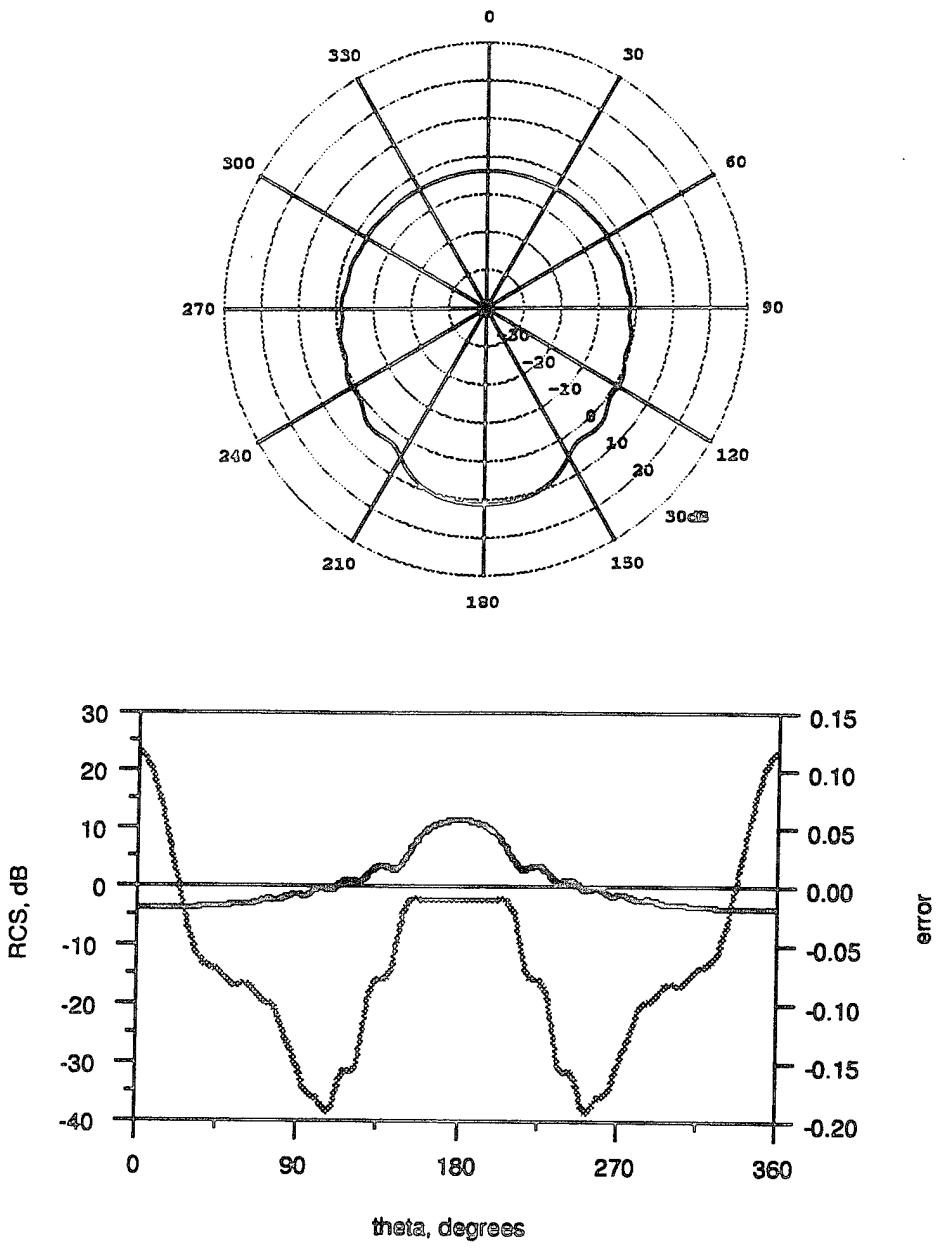

Figure 15 Cross Section for TM Scattering for Problem A.1. The light curve in the Cartesian plot is the estimated error in the result.

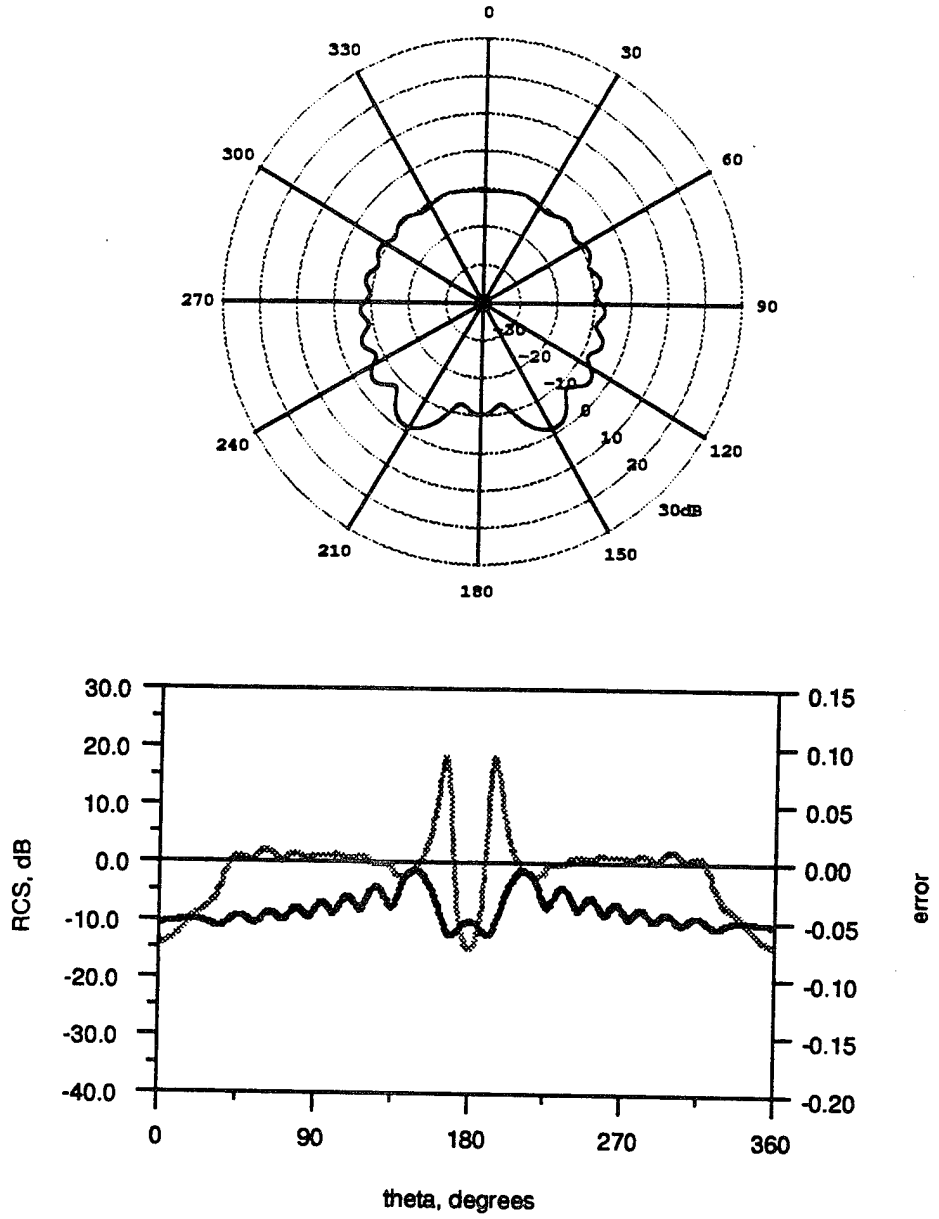

Figure 16 Cross Section for TE Scattering for Problem A.1. The light curve in the Cartesian plot is the estimated error in the result.

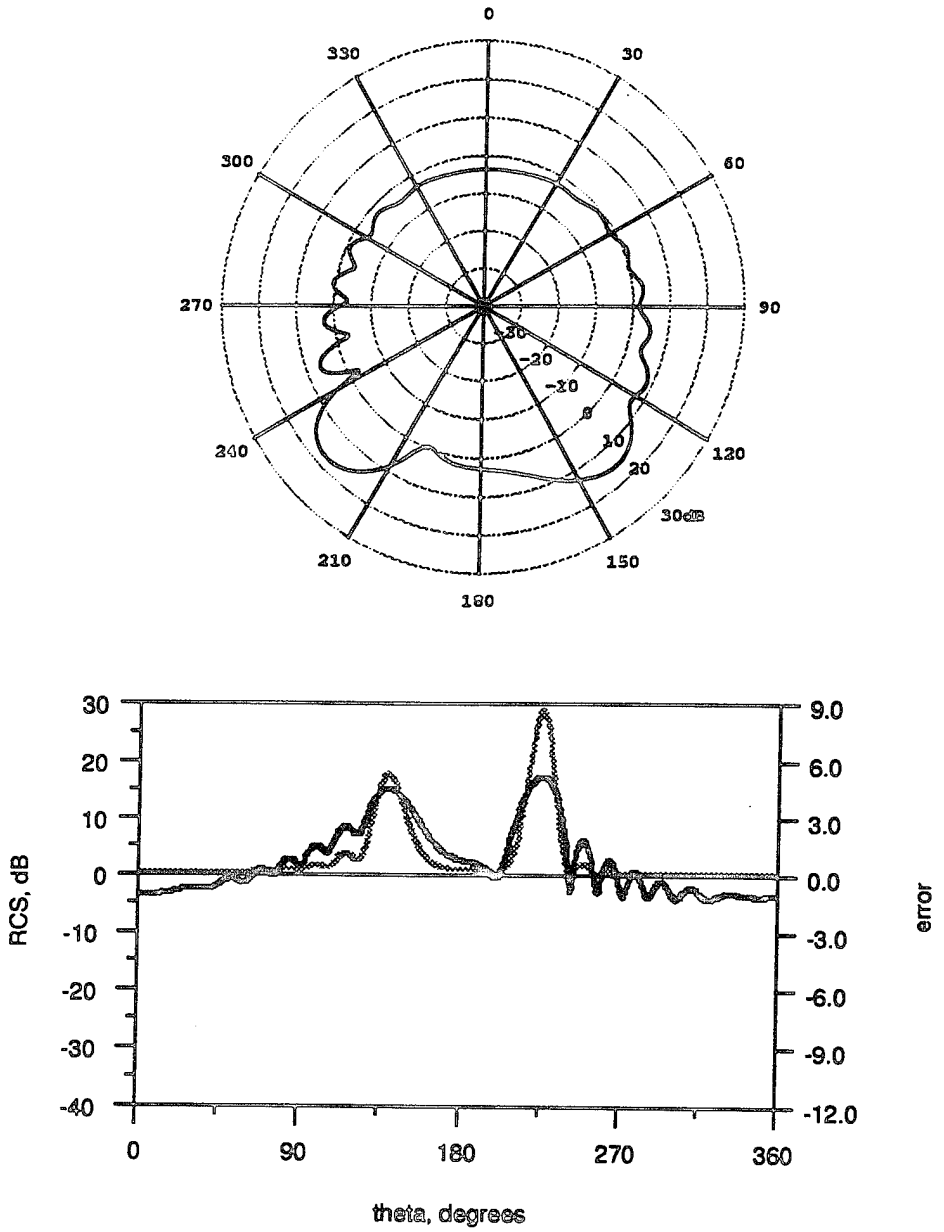

Figure 17 Cross Section for TM Scattering for Problem A.3. The light curve in the Cartesian plot is the estimated error in the result.

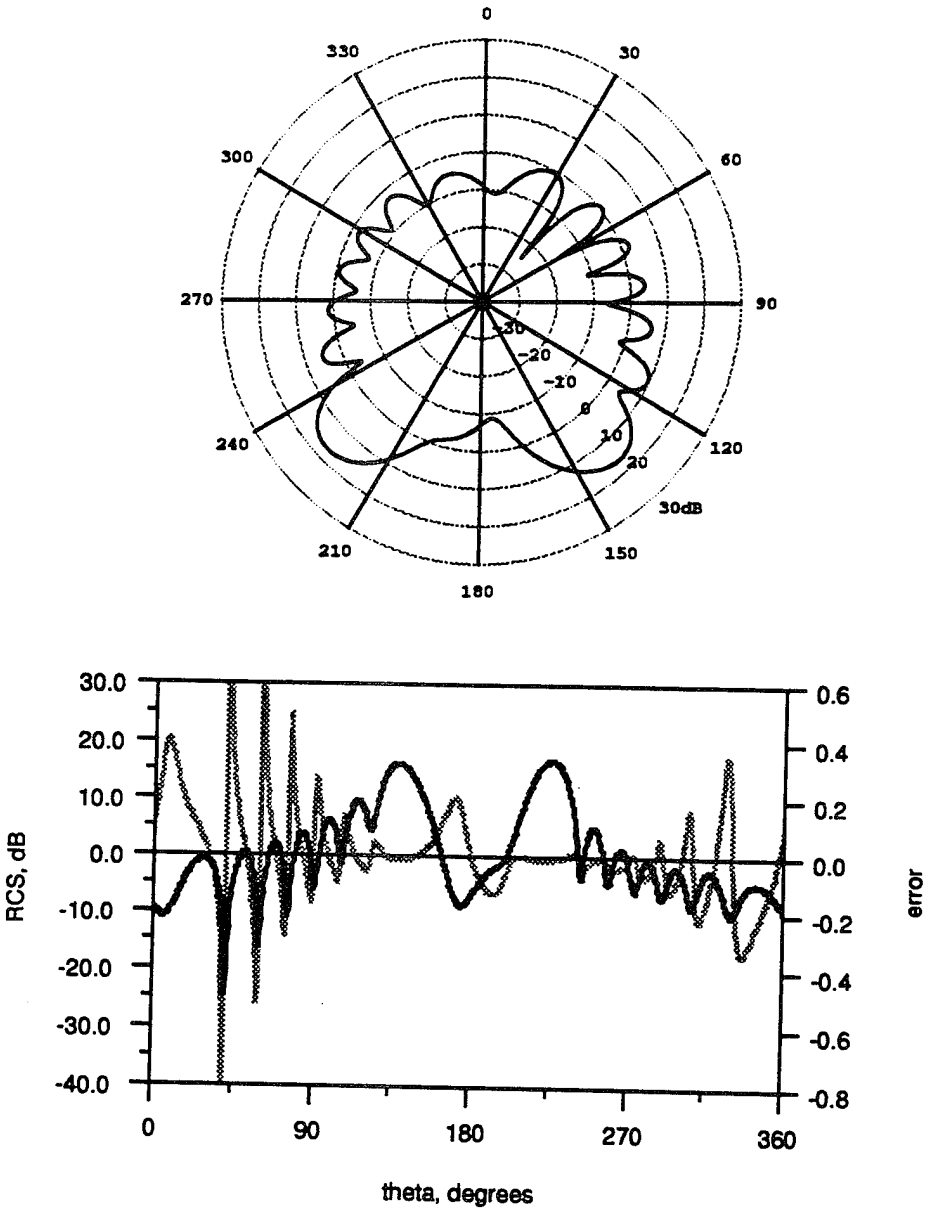

Figure 18 Cross Section for TE Scattering for Problem A.3. The light curve in the Cartesian plot is the estimated error in the result.

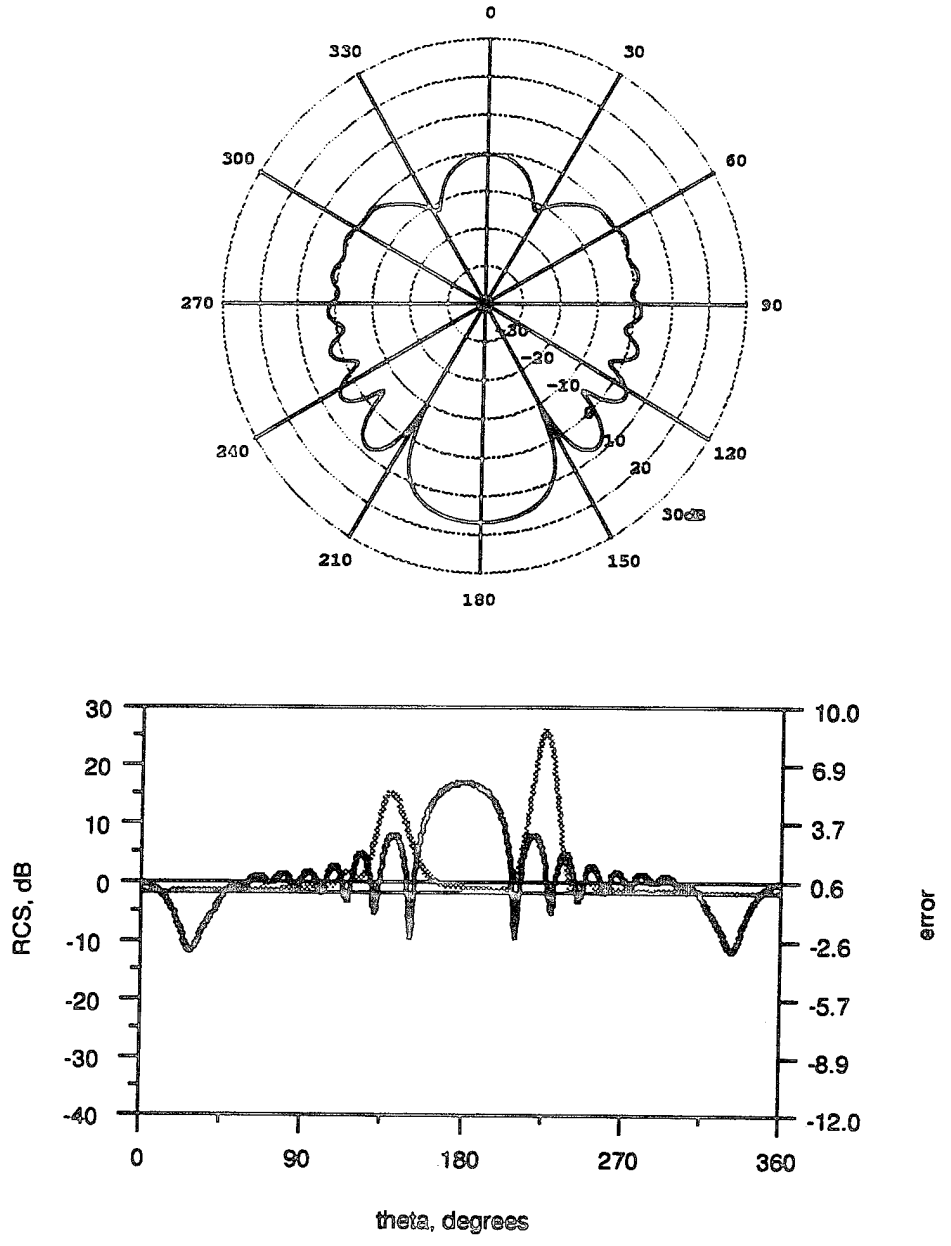

Figure 19 Cross Section for TM Scattering for Problem A.5. The light curve in the Cartesian plot is the estimated error in the result.

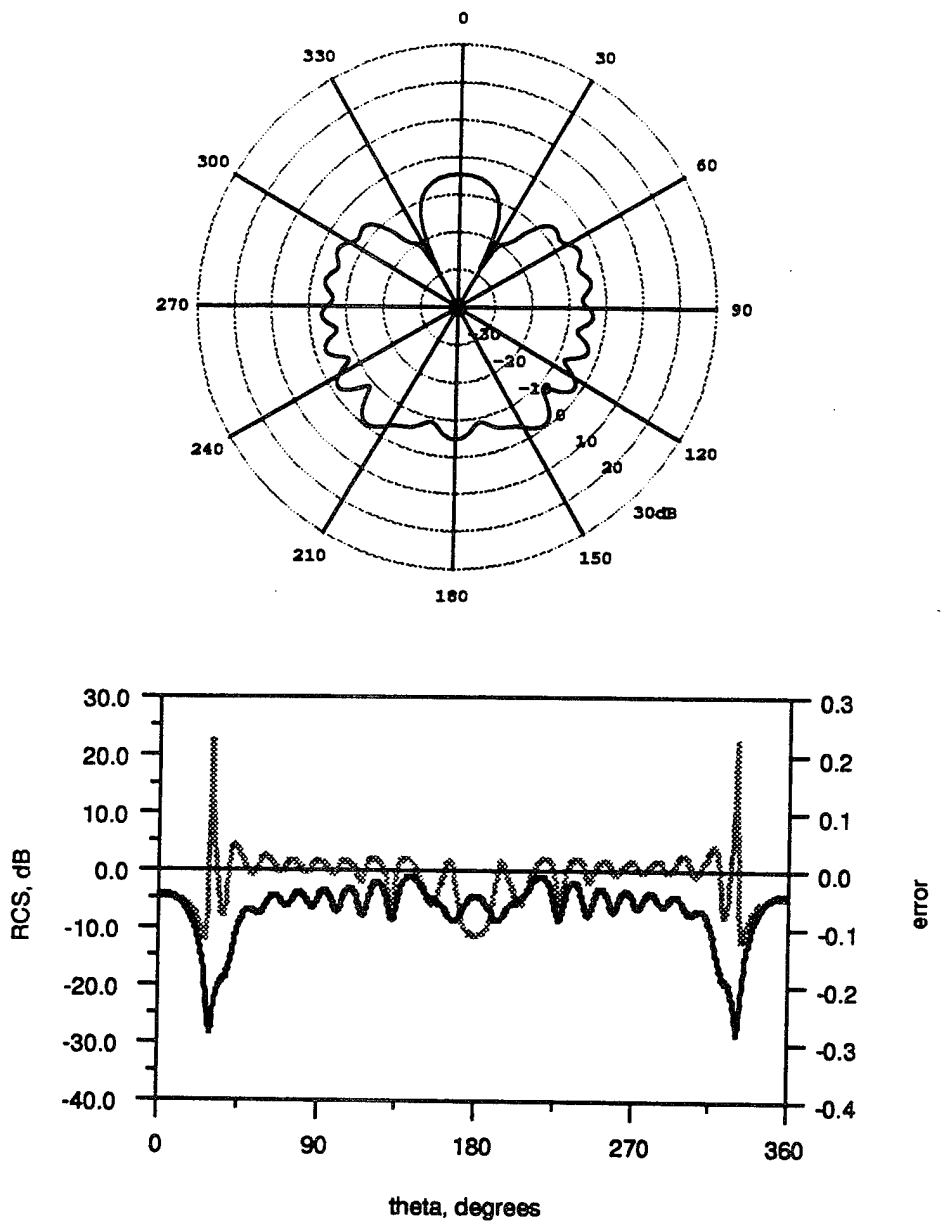

Figure 20 Cross Section for TE Scattering for Problem A.5. The light curve in the Cartesian plot is the estimated error in the result.

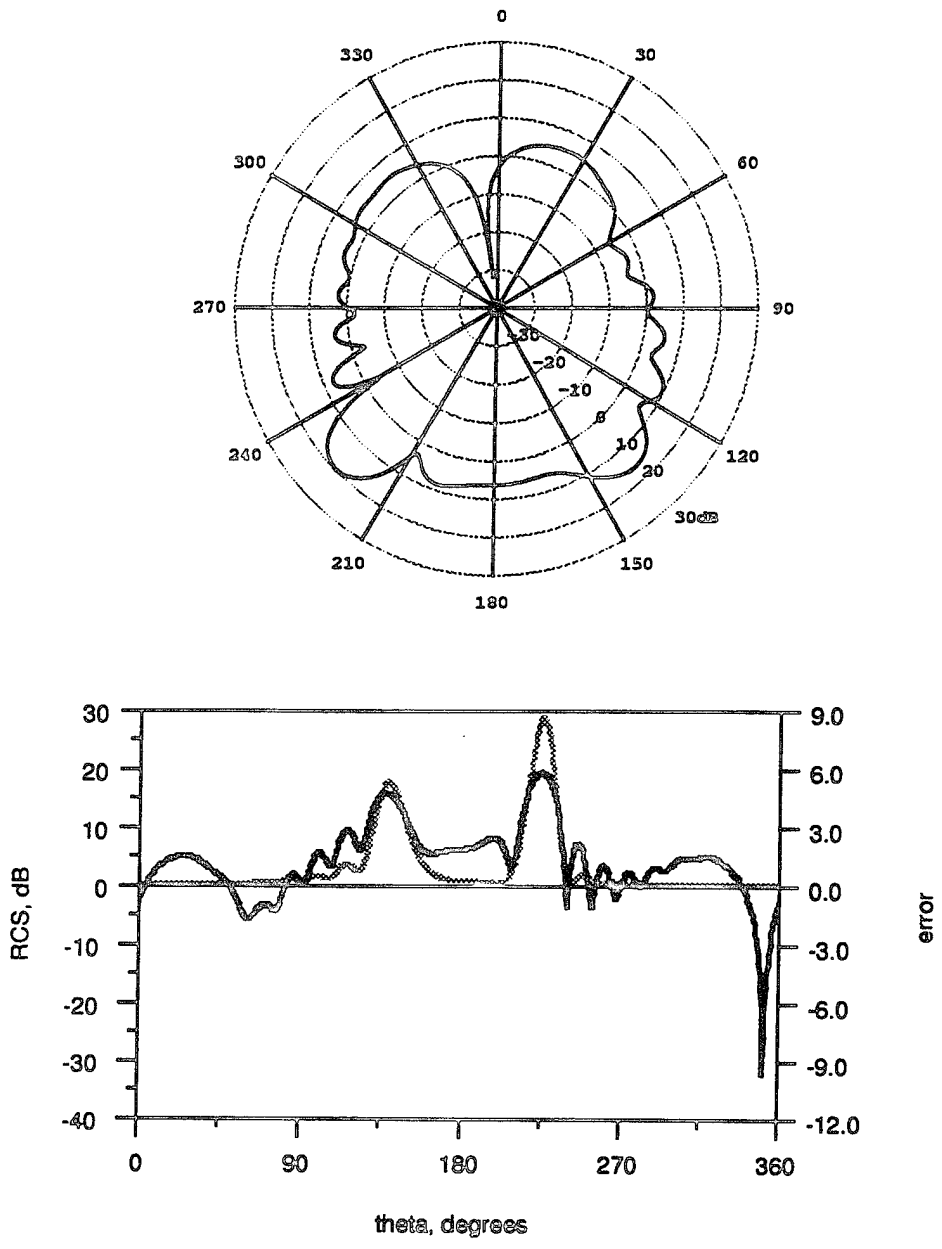

Figure 21 Cross Section for TM Scattering for Problem A.7. The light curve in the Cartesian plot is the estimated error in the result.

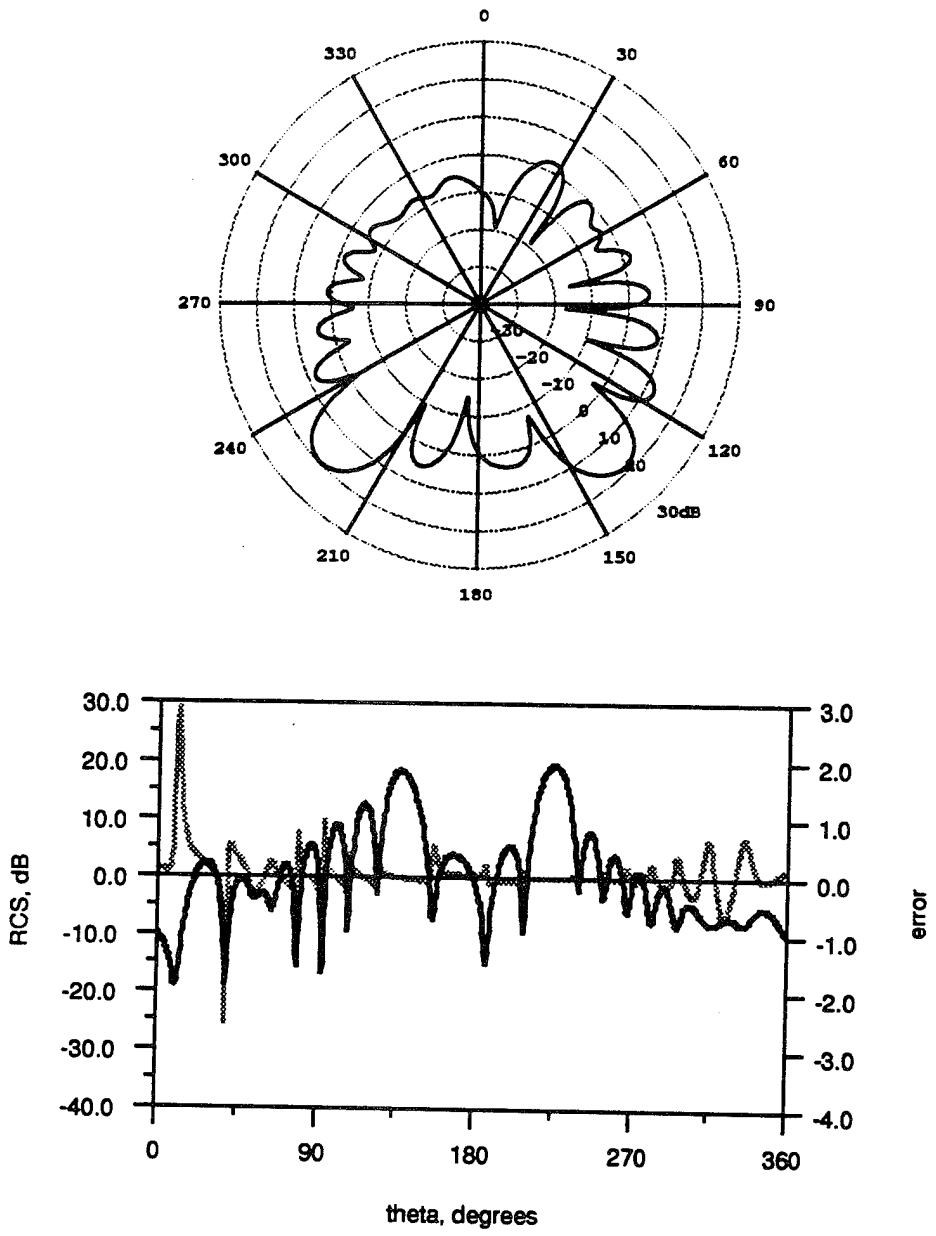

Figure 22 Cross Section for TE Scattering for Problem A.7. The light curve in the Cartesian plot is the estimated error in the result.

CAVITIES

The patch sizes on the cavities were of equal length except near the exterior corners. At the corners, the same kind of tapering was done as for the ogive and airfoils. Figure 23 shows the cavity for problems C.1 and C.2 with a single taper at each exterior corner.

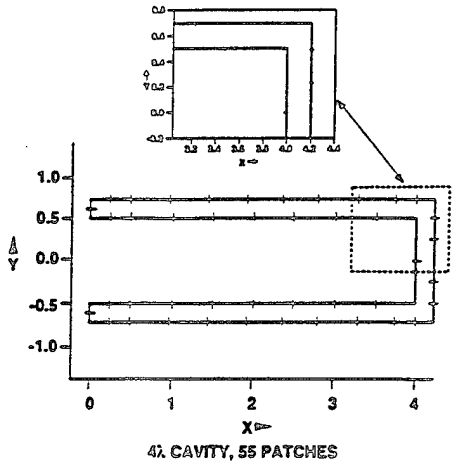

Figure 23 Sample discretization for problems C.1 and C.2.

Figure 24 is the cavity for problem C.3 with each corner patch subdivided twice.

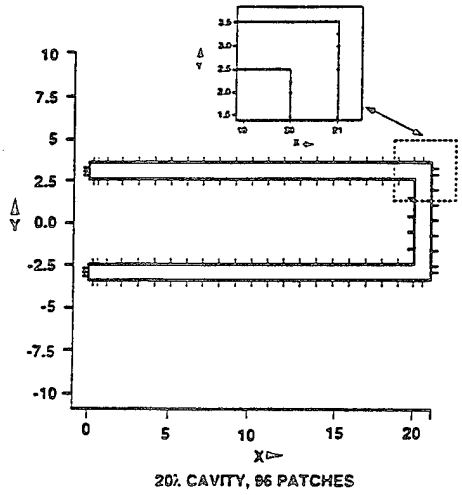

Figure 24 Sample discretization for problem C.3.

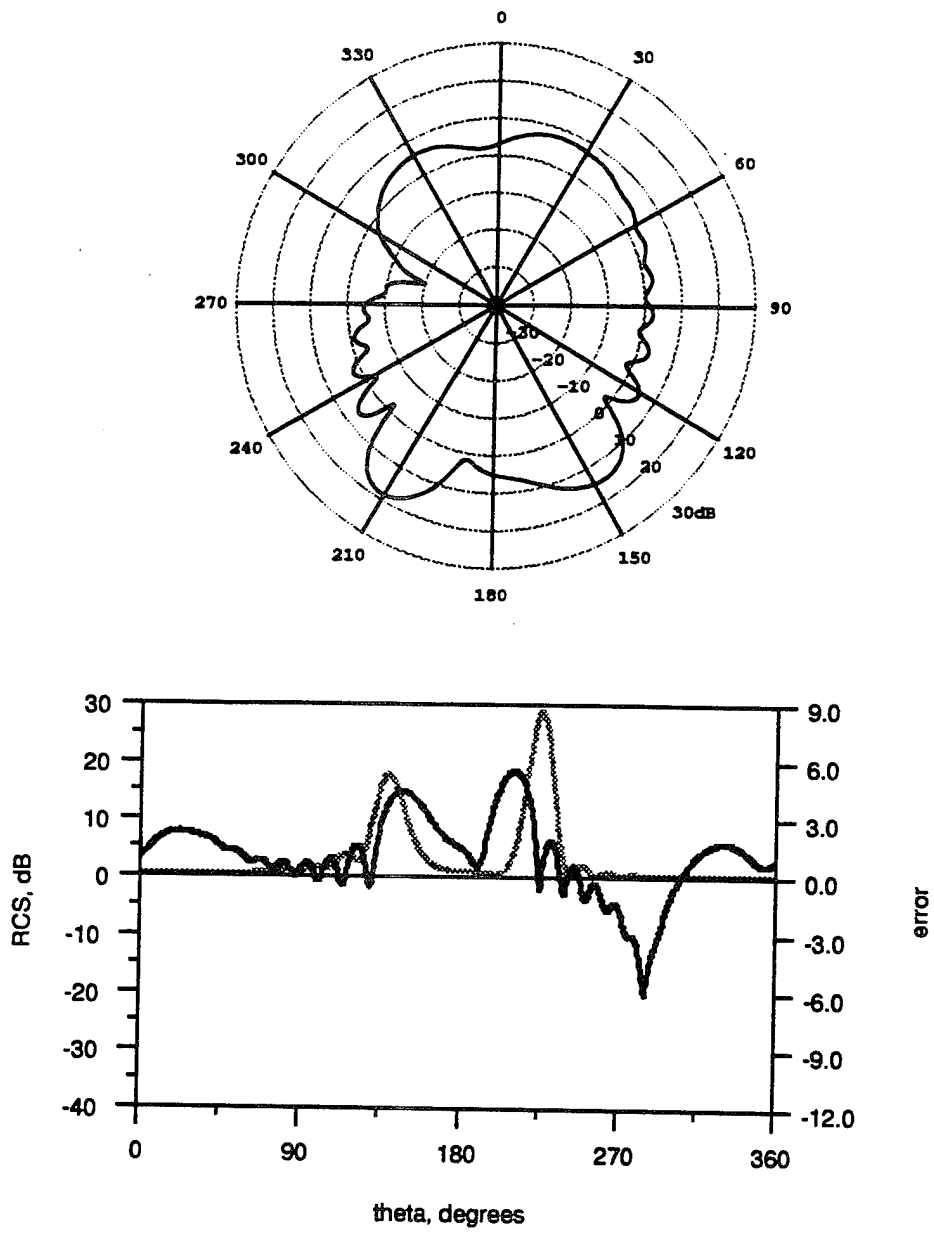

Figure 25 Cross Section for TM Scattering for Problem C.1. The light curve in the Cartesian plot is the estimated error in the result.

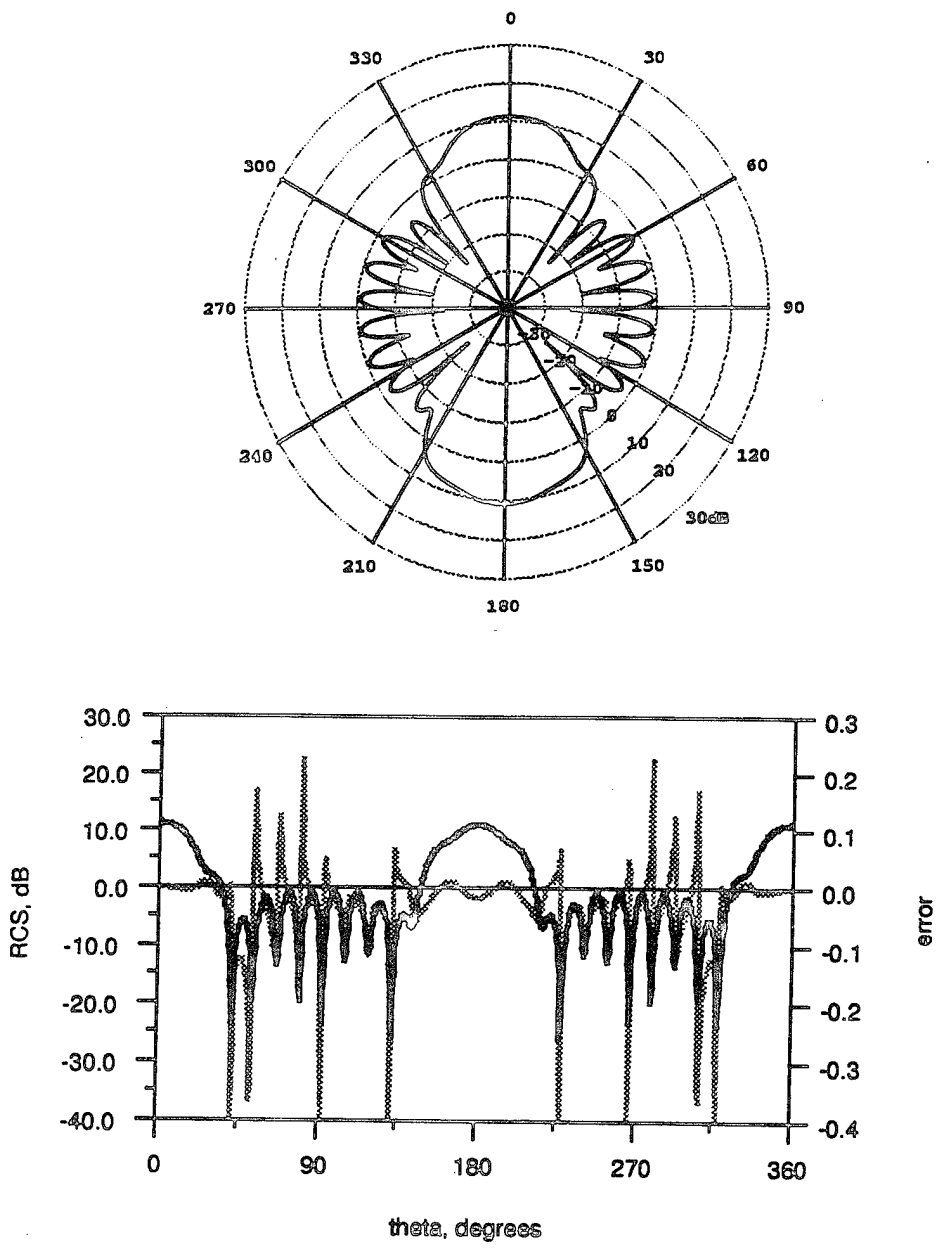

Figure 26 Cross Section for TE Scattering for Problem C.1. The light curve in the Cartesian plot is the estimated error in the result.

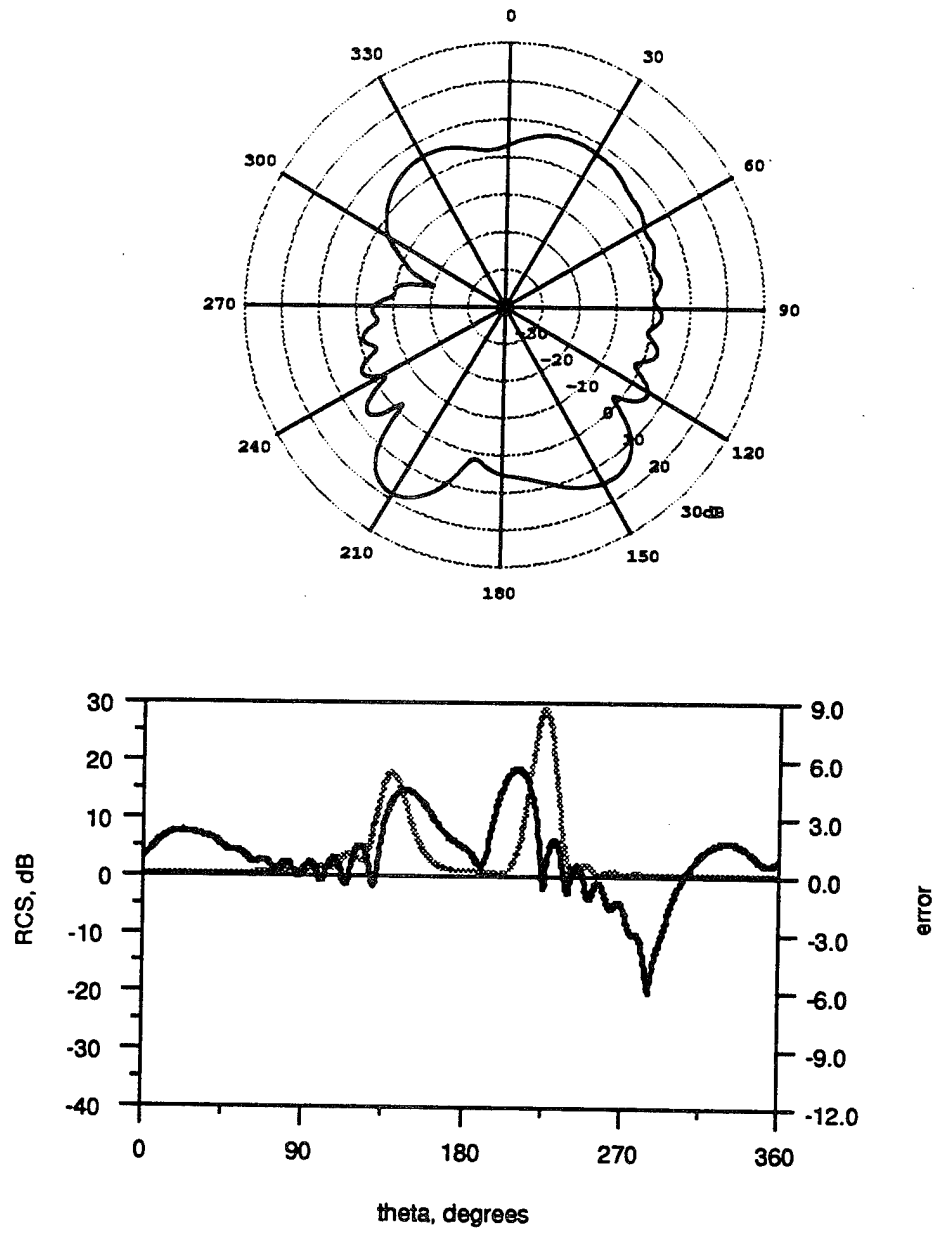

Figure 27 Cross Section for TM Scattering for Problem C.2. The light curve in the Cartesian plot is the estimated error in the result.

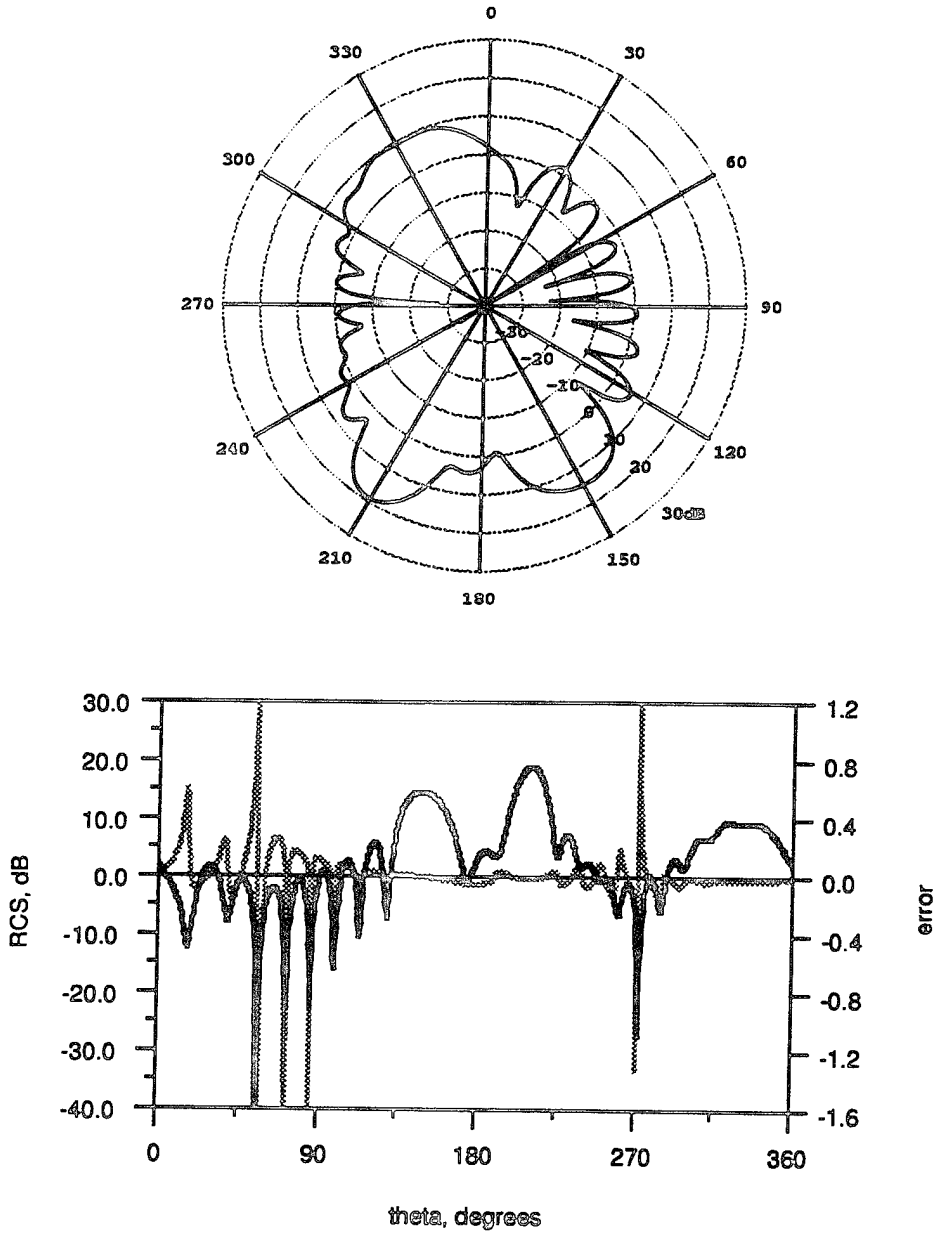

Figure 28 Cross Section for TE Scattering for Problem C.2. The light curve in the Cartesian plot is the estimated error in the result.

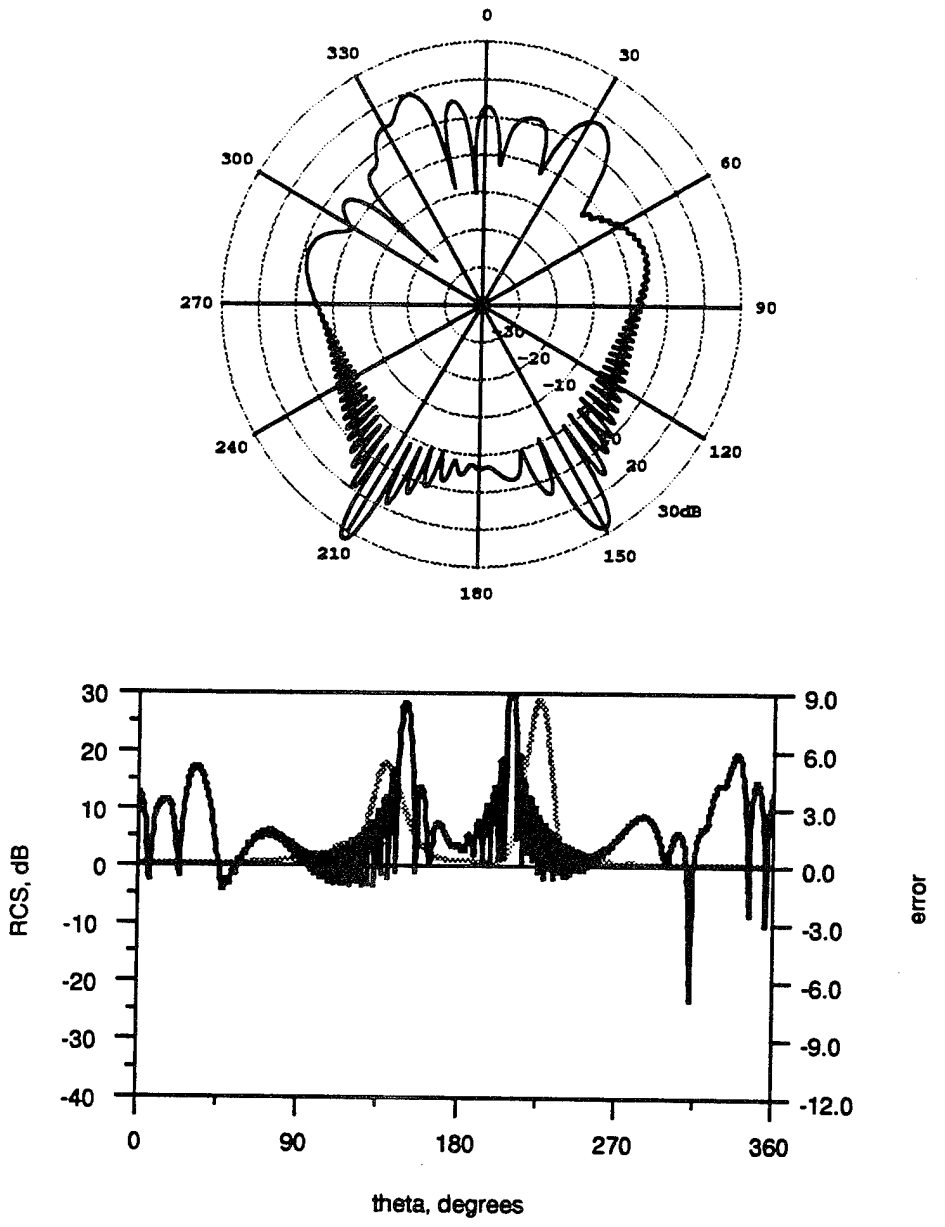

Figure 29 Cross Section for TM Scattering for Problem C.3. The light curve in the Cartesian plot is the estimated error in the result.

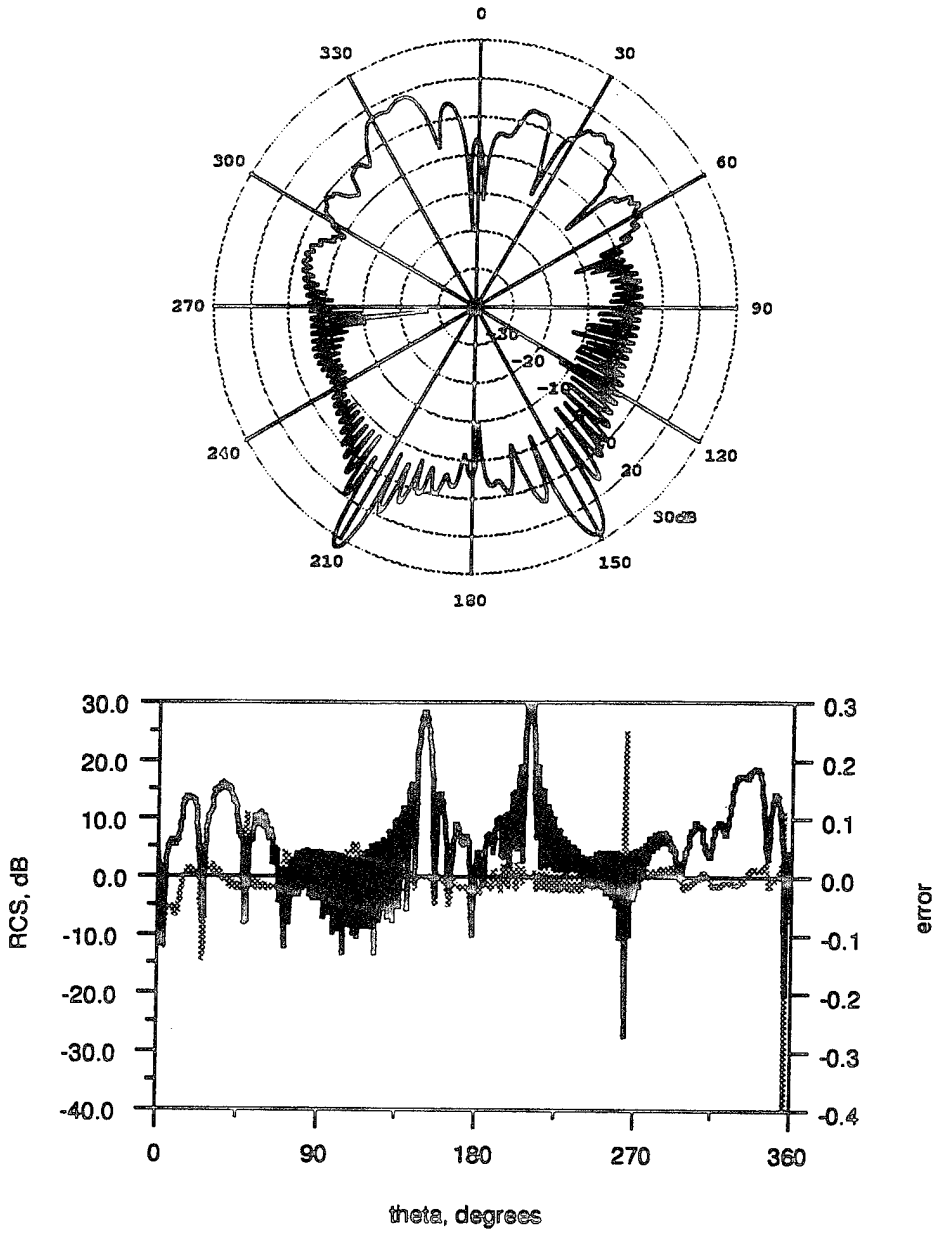

Figure 30 Cross Section for TE Scattering for Problem C.3. The light curve in the Cartesian plot is the estimated error in the result.

ANALYSIS OF RESULTS

In the tables for each problem we have listed the discretization, rms errors, run times, and accuracies. The column labeled "segments" is the number of patches on the scatterer. The "b.f. order" column is the highest order of the basis functions on each patch. The "rms error" column is the rms error divided by the average cross section. The error was computed assuming a solution we estimated to have 10 digits of accuracy to be exact. The "time" column is the CPU time in seconds on a Sun SPARCstation. The machine colunmn gives the SparcStation model. The IPX and LX models have about the same speed. The Sparc-10 is 2–3 times faster than the IPX and LX. The IPC is about half the speed of the IPX and LX.

Table 1 is a summary of the results for scattering from the ellipse in problem E.1. Note that little additional time is required to get high accuracy results compared to low accuracy results. The largest errors were at 0 degrees where the cross section is the smallest.

Table 1 E.1: Run Times, Discretization, and Accuracy

polarization	segments	b.f. order	rms error	time	machine
TM	8	3	7e-2	6.9	IPX
TM	10	4	3e-3	9.7	IPX
TM	10	9	3e-9	26.7	IPX
TE	12	2	2.5e-1	11.0	LX
TE	8	7	3.0e-3	20.0	LX
TE	10	11	3e-9	57.0	LX

The data for problem E.3 was similar to, but slightly better than, the E.1 results. The largest errors in the TM cross section were near 0 degrees. The largest errors in the TE cross section were at the nulls near 200 and 270 degrees.

Table 2 is a summary of the results for scattering from the ellipse in problem E.4. The largest errors were near 0 degrees and 180 degrees. The results for problem E.6 were similar.

Table 2 E.6: Run Times, Discretization, and Accuracy

polarization	segments	b.f. order	rms error	time	machine
TM	30	4	8e-2	34.1	IPX
TM	40	4	1e-2	52.5	IPX
TM	38	10	1e-8	195	IPX
TE	30	5	1.3e-1	58.9	IPX
TE	40	7	2.8e-3	147	LX
TE	20	18	1.6e-8	388	IPX

Table 3 is a summary of the results for scattering from the ogive in problem O.1. The errors were approximately the same in each angular region. The need to taper the segments many times near the tips caused the run times for extremely high accuracy solutions to be much longer than the lower accuracy ones.

Table 3 O.1: Run Times, Discretization, and Accuracy

polarization	taper	b.f. order	rms error	time	machine
TM	1	0	1e-1	4.2	IPX
TM	2	1	3e-3	6.8	IPX
TM	15	5	1e-8	136	IPX
TE	1	1	1.2e-1	6.9	IPX
TE	2	2	9e-3	11	IPX
TE	6	16	1e-8	245	IPX

Table 4 is a summary of the results for scattering from the single airfoil in problem A.1. The largest errors were where the cross section was smallest: near 0 degrees in the TM case and near 0 degrees and 180 degrees in the TE case.

Table 4 A.1: Run Times, Discretization, and Accuracy

polarization	segments	b.f. order	taper	rms error	time	machine
TM	12	1	2	1e-1	9	IPX
TM	10	3	2	1e-2	14	IPX
TM	26	8	14	9e-9	211	IPX
TE	10	4	0	1.0e-1	14	LX
TE	10	6	0	4.6e-3	21	LX
TE	26	10	14	8.6e-9	422	IPX

The discretizations and run times for problem A.3 were simliar to those for A.1. The errors in the cross section for the TM case were the largest near the regions of maximum cross section near 140 and 230 degrees. There was a large error spike in the TE case at the deep null at 40 degrees.

Table 5 is a summary of the results for scattering from the double airfoil in problem A.5. The results for problem A.7 are quite similar. The discretizations were almost the same as in problems A.1 and A.3 on each air foil. For accuracies of 2 digits, the errors in both these problems were largest near 140 degrees and 220 degrees. In the higher accuracy results, there were relatively large error spikes where the cross section was lowest.

Table 5 A.5: Run Times, Discretization, and Accuracy

polarization	segments	b.f. order	taper	rms error	time	machine
TM	12	2	2	7e-2	23	IPX
TM	10	3	2	2e-2	25	IPX
TM	26	8	14	3e-8	1553	IPX
TE	10	4	0	1.4e-1	26	IPX
TE	10	6	0	8.6e-3	40	IPX
TE	26	11	15	4.3e-9	2343	10

Table 6 is a summary of the results for scattering from the small cavity in problem C.1. The errors tended to be the largest in the regions where the cross section was lowest.

Table 6 C.1: Run Times, Discretization, and Accuracy

polarization	segments	b.f. order	taper	rms error	time	machine
TM	24	1	1	1.4e-1	18	IPX
TM	24	1	2	1.4e-2	27	IPX
TM	24	12	7	1.1e-8	2056	IPX
TE	24	1	3	5.4e-2	40	LX
TE	24	1	4	3.6e-3	55	LX
TE	24	13	9	5.8e-9	1467	10

Table 7 is a summary of the results for scattering from the small cavity in problem C.2. The errors tended to be the largest in the regions where the cross section was lowest.

Table 7 C.2: Run Times, Discretization, and Accuracy

polarization	segments	b.f. order	taper	rms error	time	machine
TM	26	1	1	8.4e-2	32	IPC
TM	24	1	2	9.9e-3	48	IPC
TM	24	12	7	1.1e-8	4048	IPX
TE	24	1	3	5.3e-2	41	LX
TE	24	1	4	7.3e-3	56	LX
TE	24	13	9	5.5e-9	4372	IPX

Table 8 is a summary of the results for scattering from the large cavity in problem C.3. The errors tended to be the largest in the same places as the other two cavities.

Table 8 C.3: Run Times, Discretization, and Accuracy

polarization	segments	b.f. order	taper	rms error	time	machine
TM	180	1	1	1.4e-1	178	LX
TM	140	1	2	1.4e-2	187	LX
TM	120	10	7	1.3e-7	1765	10
TE	120	1	3	1.1e-1	260	LX
TE	120	1	5	1.6e-3	401	IPX
TE	120	13	9	4.6e-9	3743	10

CONCLUSIONS

The most striking conclusion from our data is that the use of accurate surface descriptions, high order basis functions, and careful quadratures permits high precision radar cross sections at a modest cost. Because it is not that much more difficult to compute a result with 3 or 4 digits of accuracy than it is to compute results with 1 or 2 digits of accuracy, one can easily check whether the solutions have actually converged.

To get two digits of precision in the cross sections, a good rule of thumb seems to be to use about 2 patches per wavelength with 3rd or 4th order basis functions on each patch. Near sharp edges, the patches should be tapered down to about 8 patches per

wavelength.

There is considerably more structure in the TE cross sections than in the TM cross sections. In spite of this, the same discretizations seem to be appropriate in each case for modest accuracies.

The cross section computations tended to have the largest errors in the regions where the cross sections were lowest. The exceptions to this rule were most frequent in 1–2 digit accuracy solutions with TM scattering.

REFERENCES

Hamilton L., Rokhlin V., Stalzer M., Turley R.S., Visher J. and Wandzura S. 1993. The Importance of Accurate Surface Models in RCS Computations. In: *IEEE Antennas and Propagation Society Symposium Digest*, pp. 1136–1139.

Hamilton L., Stalzer M., Turley R.S., Visher J. and Wandzura S. 1993a. Method of Moments Scattering Computations Using High-Order Basis Functions. In: *IEEE Antennas and Propagation Society Symposium Digest*, pp. 1132–1135.

Ma J., Rokhlin V. and Wandzura W. 1993. Generalized Gaussian Quadrature Rules for Systems of Arbitrary Functions. Yale University Research Report YALEU/DCS/RR-990.

Rao S.M., Wilton D.R. and Glisson A.W. 1982. Electromagnetic Scattering by Surfaces of Arbitrary Shape. *IEEE Transactions on Antennas and Propagation*, AP-30(3): 409–418.

Wandzura S.M. 1991. Optimality of Galerkin Method of Scattering Computations. *Microwave and Optical Technology Letters*, 4(5):199–200.

Appendix C

Continuous Basis Functions

This is a *Mathematica* notebook that defines and illustrates how high-order polynomial basis functions with continuity properties appropriate for discretization of some $2d$ and $3d$ integral equations may be constructed. It is bound separately and included with selected copies of this report.

Appendix D

Scalable Quadratures

This is a *Mathematica* notebook that defines and tablulates the quadrature rules used in FastScat. All the rules (other than Gauss-Legendre and the low-order triangle rules) have be derived under this contract. It is bound separately and included with selected copies of this report.

Bibliography

[ABB+92] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J.Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A.McKenney, S. Ostouchov, and D. Sorensen. *LAPACK User's Guide.* Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[AS72] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions.* Applied Mathematics Series. National Bureau of Standards, Cambridge, 1972.

[Bro75] Frederick P. Brooks, Jr. *The Mythical Man-Month.* Addison-Wesley, Reading, Massachusetts, 1975.

[Che90] Weng C. Chew. A derivation of the vector addition theorem. *Microwave and Optical Technology Letters*, 3(7):256–260, 1990.

[Che92] Weng C. Chew. Recurrence relations for three-dimensional scalar addition theorem. *Journal of Electromagnetic Waves and Applications*, 6(2):133–142, 1992.

[CRW93a] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. The fast multipole method: A pedestrian prescription. *IEEE Antennas and Propagation Society Magazine*, 35(3):7–12, June 1993.

[CRW93b] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. The fast multipole method for electromagnetic scattering calculations. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 1, pages 48–51, Ann Arbor, MI, June 1993. IEEE.

[CRW93c] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. The fast multipole method for pedestrians. In *Progress In Electromagnetics Research Symposium*, Pasadena, CA, July 1993. Jet Propulsion Laboratory.

[CRW94] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. Faster single-stage multipole method for the wave equation. In *10th An-*

nual Review of Progress in Applied Computational Electromagnetics, volume 1, pages 19–24, Monterey, CA, March 1994. Applied Computational Electromagnetics Society.

[DCDH90] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Soft.*, 16:1–17, 1990.

[DCHH88] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. Algorithm 656: An extended set of FORTRAN Basic Linear Algebra Subprograms. *ACM Trans. Math. Soft.*, 14:1–17, 1988.

[DY94] B. Dembart and Elizabeth Yip. A 3D moment method code based on fast multipole. In *URSI Radio Science Meeting*, page 23, Seattle, WA, June 1994. URSI.

[DY95] Benjamin Dembart and Elizabeth Yip. A 3D fast multipole method for electromagnetics with multiple levels. In *11th Annual Review of Progress in Applied Computational Electromagnetics*, volume 1, pages 621–628, Monterey, CA, March 1995. Applied Computational Electromagnetics Society.

[ED95] Michael A. Epton and Benjamin Dembart. Multipole translation theory for the three-dimensional laplace and helmholtz equations. *SIAM Journal of Scientific Computing*, 16(4):865–897, July 1995.

[ES90] Margaret A. Ellis and Bjarne Stroustrup. *The Annotated C++ Reference Manual.* Addison-Wesley, Reading, Massachusetts, 1990.

[Fey49] Richard P. Feynman. Space-time approach to quantum electrodynamics. *Physical Review*, 76:769–789, 1949.

[HMS+94a] Lisa R. Hamilton, Perry A. Macdonald, Mark A. Stalzer, R. Steven Turley, John L. Visher, and Stephen M. Wandzura. Electromagnetic scatering computations using high-order basis functions in the method of moments. In *IEEE Antennas and Propagation Society International Symposium Digest*, volume 3, pages 2166–2169, Seattle, WA, June 1994. IEEE.

[HMS+94b] Lisa R. Hamilton, Perry A. Macdonald, Mark A. Stalzer, R. Steven Turley, John L. Visher, and Stephen M. Wandzura. 3D method of moments scattering computations using the fast multipole method. In *IEEE Antennas and Propagation Society International Symposium Digest*, volume 1, pages 435–438, Seattle, WA, June 1994. IEEE.

[HOS+95a] Lisa R. Hamilton, John J. Ottusch, Mark A. Stalzer, R. Steven Turley, John L. Visher, and Stephen M. Wandzura. Fastscat benchmark data. In *Proc. 1994 HAVE FORUM Symposium*, volume I, pages 255–268, Wright Patterson AFB, OH 454-7523, February 1995. Wright Laboratory. WL–TR–95–6003.

[HOS+95b] Lisa R. Hamilton, John J. Ottusch, Mark A. Stalzer, R. Steven Turley, John L. Visher, and Stephen M. Wandzura. Accuracy estimation and high order methods. In *11th Annual Review of Progress in Applied Computational Electromagnetics*, volume II, pages 1177–1184, Monterey, CA, March 1995. Applied Computational Electromagnetics Society.

[HOS+95c] Lisa R. Hamilton, John J. Ottusch, Mark A. Stalzer, R. Steven Turley, John L. Visher, and Stephen M. Wandzura. Fast multipole methods for scattering computation. Annual contract report, Hughes Aircraft Company Research Laboratories, February 1995. AFOSR Contract No. F49620-91-C-0064.

[HRS+93] Lisa Hamilton, Vladimir Rokhlin, Mark Stalzer, R. Steven Turley, John Visher, and Stephen Wandzura. The importance of accurate surface models in RCS computations. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 3, pages 1136–1139, Ann Arbor, MI, June 1993. IEEE.

[HST+93a] Lisa Hamilton, Mark Stalzer, R. Steven Turley, John Visher, and Stephen Wandzura. FastScat: an object-oriented program for fast scattering computation. *Scientific Programming*, 2(4):171–178, 1993.

[HST+93b] Lisa Hamilton, Mark Stalzer, R. Steven Turley, John Visher, and Stephen Wandzura. Method of moments scattering computations using high-order basis functions. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 3, pages 1132–1135, Ann Arbor, MI, June 1993. IEEE.

[HST+93c] Lisa R. Hamilton, Mark A. Stalzer, R. Steven Turley, John L. Visher, and Stephen M. Wandzura. Fast multipole methods for scattering computation. Annual contract report, Hughes Aircraft Company Research Laboratories, August 1993. AFOSR Contract No. F49620-91-C-0064.

[HST+94] Lisa Hamilton, Mark A. Stalzer, R. Steven Turley, John L. Visher, and Stephen Wandzura. Surface modeling in C++. In *Proceedings of the Second Annual Object-Oriented Numerics Conference*, pages 50–59, Corvallis, OR, 1994. Rogue Wave Software.

[JWS88] W. A. Johnson, Donald R. Wilton, and R. M. Sharpe. Patch code users' manual. Sandia Report SAND87–2991, Sandia National Laboratories, Albuquerque, New Mexico, May 1988.

[KGV83] Scott Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671, 1983.

[Kir84] Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34:975, 1984.

[KTW93] Winifred Kummer, R. Steven Turley, and Stephen Wandzura. Application of the fast Fourier transform to infinite periodic structures. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 3, pages 1246–1249, Ann Arbor, MI, June 1993. IEEE.

[LC95] Cai-Cheng Lu and Weng Cho Chew. Fast far field approximation for calculating the rcs of large objects. In *IEEE Antennas and Propagation Society International Symposium Digest*, volume 1, pages 22–25, Newport Beach, CA, June 1995. IEEE.

[LSL87] S. Lee, D.A. Shnidman, and F.A. Lichauco. Numerical modeling of RCS and antenna problems. Technical Report ESD-TR-87-935, Massachusetts Institute of Technology Lincoln Laboratories, 1987. National Security Agency Electronic System Division Contract F19628-85-C-0002.

[Mey88] Bertrand Meyer. *Object-Oriented Software Construction.* Prentice Hall, New York, 1988.

[MRW93] Jin-Hong Ma, Vladimir Rokhlin, and Stephen Wandzura. Generalized gaussian quadrature rules for systems of arbitrary functions. Technical Report YALEU/DCS/RR-990, Yale University, Department of Computer Science, October 1993. To be published in SIAM Journal of Numerical Analysis.

[Ott94] John J. Ottusch. Performance comparison of FastScat(TM) and RAM2D. In *Electromagnetic Code Consortium*, Albuquerque, NM, May 1994.

[PMMG92] J. M. Putnam, L. N. Medgyesi-Mitschang, and M. B. Gedera. Carlos-3d(tm) three-dimensional method of moments code, theory and user manual. Technical report, McDonnell Douglas Corporation, St. Louis, MO, Dec 1992.

[PV49] Wolfgang Pauli and F. Villars. On the invariant regularization in relativistic quantum theory. *Reviews of Modern Physics*, 21:434–444, 1949.

[Rok90] Vladimir Rokhlin. Rapid solution of integral equations of scattering theory in two dimensions. *Journal of Computational Physics*, 86(2):414–439, 1990.

[Rok93] Vladimir Rokhlin. Diagonal form of translation operators for the Helmholtz equation in three dimensions. *Applied and Computational Harmonic Analysis*, 1(1):82–93, December 1993.

[RW94] Vladimir Rokhlin and Stephen Wandzura. The fast multipole method for periodic structures. In *IEEE Antennas and Propagation Society International Symposium Digest*, volume 1, pages 424–426, Seattle, WA, June 1994. IEEE.

[SC95] Jiming M. Song and Weng Cho Chew. Fast multipole method solution of three dimensional integral equation. In *IEEE Antennas and Propagation Society International Symposium Digest*, volume 3, pages 1528–1531, Newport Beach, CA, June 1995. IEEE.

[Sta95] Mark A. Stalzer. A parallel fast multipole method for the Helmholtz equation. To be published in *Parallel Processing Letters*, 1995.

[Str94] John Strain. Locally-corrected multidimensional quadrature rules for singular functions. Technical report, Lawrence Berkeley Laboratory, 1994. To be published in *SIAM Journal of Scientific Computing*.

[Wan92] Stephen M. Wandzura. Electric current basis functions for curved surfaces. *Electromagnetics*, 12:77–91, 1992.

[Wan95a] Stephen Wandzura. High-order discretization of integral equations with singular kernels. In *IEEE Antennas and Propagation Society International Symposium Digest*, volume 1, pages 792–795, Newport Beach, CA, June 1995. IEEE.

[Wan95b] Stephen M. Wandzura. Accuracy in computation of matrix elements of singular kernels. In *11th Annual Review of Progress in Applied Computational Electromagnetics*, volume II, pages 1170–1176, Monterey, CA, March 1995. Applied Computational Electromagnetics Society.

[WC93] Y. M. Wang and Weng Cho Chew. An efficient way to compute the vector addition theorem. In *IEEE Antennas and Propagation Society Symposium Digest*, volume 1, pages 174–177, Ann Arbor, MI, June 1993. IEEE.

[WC94] Robert L. Wagner and Weng C. Chew. A ray-propagation fast multipole algorithm. *Microwave and Optical Technology Letters*, 7(10):435–438, July 1994.

[WSWS92] Helen T. G. Wang, Michael L. Sanders, Alex Woo, and Michael Schuh. Radar cross section measurement data EMCC benchmark targets (RAM coated and dielectric targets). Technical Memorandum NAWCWPNS TM 7318, Naval Air Warfare Center Weapons Division, China Lake, August 1992. Limited Distribution Document.